www.GossamerSec.com

Assurance Activity Report for Samsung Electronics Co., Ltd. Samsung Galaxy Devices on Android 14 - Fall

Version 0.2
12/16/2024

Prepared by:
Gossamer Security Solutions
Accredited Security Testing Laboratory – Common Criteria Testing
Columbia, MD 21045

Prepared for:
National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

Revision History

Revision	Date	Authors	Summary
Version 0.1	11/08/2024	Gossamer	Initial draft
Version 0.2	12/16/2024	Gossamer	Addressed ECR comments

The TOE Evaluation was Sponsored by:

Samsung Electronics Co., Ltd.
416 Maetan-3dong, Yeongtong-gu, Suwon-si, Gyeonggi-do, 443-742 Korea

Evaluation Personnel:

- James Arnold
- Tammy Compton
- Rizheng Sun

Common Criteria Versions:

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017

Common Evaluation Methodology Versions:

- Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017

Table of Contents

1. Introduction

This document presents evaluations results of the Samsung Electronics Co., Ltd. Samsung Galaxy Devices on Android 14 – Fall MDFPP33/BIO11/WLANC10/PKGTLS11/BT10/VPNC24 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators. All references to requirements are meant to be within the AAR unless ST is specifically referenced.

1.1 Device Equivalence

This evaluation tested the following Galaxy devices. All devices were running Android 14.

Device Name	Chipset Vendor	SoC	Arch	Kernel	Build Number
Galaxy Z Flip6 5G	Qualcomm	Snapdragon 8 Gen 3	ARMv8	6.1	UP1A.231005.007
Galaxy A52 5G	Qualcomm	Snapdragon 750	ARMv8	4.19	UP1A.231005.007
Galaxy A53 5G	Samsung	Exynos 1280	ARMv8	5.10	UP1A.231005.007
Galaxy S24 FE	Samsung	Exynos 2400	ARMv8	6.1	UP1A.231005.007

Table 1 - Evaluated Devices

In addition to the evaluated devices, the below devices are claimed as equivalent with a note about the differences between the evaluated device (first column) and the equivalent models (noted in the third column with the differences in the fourth column). Equivalence in this table is determined by the use of identical processors, kernel and build number, and is not made across processor types.

Evaluated Device	SoC	Equivalent Devices	Differences
Galaxy Z Flip6 5G	Snapdragon 8 Gen 3	Galaxy Z Fold6 5G	Z Fold6 > Z Flip6 in terms of display size
		Galaxy Z Fold SE	Z Fold SE > Z Flip6 in terms of display size
Galaxy A52 5G	Snapdragon 750	N/A	
Galaxy A53 5G	Exynos 1280	N/A	
Galaxy S24 FE	Exynos 2400	N/A	

Table 2 - Equivalent Devices

For each device, there are specific models that are validated. This table lists the specific carrier models that have the validated configuration (covering both evaluated and equivalent devices).

Device Name	Chipset Vendor	Chipset Name	Base Model Number	Carrier Models
Galaxy Z Fold6 5G	Qualcomm	Snapdragon 8 Gen 3 (SM8650)	SM-F956	W, B, N, U1, U, Q, SC-55E, SCG28
Galaxy Z Fold SE	Qualcomm	Snapdragon 8 Gen 3 (SM8650)	SM-F958	N
Galaxy S24 FE	Samsung	Exynos 2400 (S5E9945)	SM-S721	W, B, N, U1, U, Q, SCG30
Galaxy Z Flip6 5G	Qualcomm	Snapdragon 8 Gen 3 (SM8650)	SM-F741	W, B, N, U1, U, Q, SC-54E, SCG29
Galaxy A52 5G	Samsung	Snapdragon 750G (SM7225)	SM-A526	W, B, U1, U, SC-53B
Galaxy A53 5G	Exynos	Exynos 1280 (S5E8825))	SM-A536 SM-S536	W, E, B, N, U1, V, U, SC-53C*, SCG15* DL

Table 3 – Carrier Models

1.2 CAVP Certificates

The TOE performs cryptographic algorithms in accordance with the following NIST standards and has received the following CAVP algorithm certificates.

The BoringSSL v1.8 library (with both Processor Algorithm Accelerators (PAA) and without PAA) provides the following algorithms.

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128/256 CBC, GCM, KW	FIPS 197, SP 800-38A/D/F	FCS_COP.1/ENCRYPT	A5074
CVL ECC - P-256/384/521	SP 800-56A	FCS_CKM.2/UNLOCKED FCS_CKM_EXT.3	A5074
DRBG CTR - 256	SP 800-90A	FCS_RBG_EXT.1	A5074
ECDSA PKG/PKV/SigGen/SigVer – P-256/384/521	FIPS 186-4	FCS_CKM.1 FCS_CKM.2/UNLOCKED FCS_COP.1/SIGN	A5074
HMAC SHA-1/256/384/512	FIPS 198-1 & 180-4	FCS_COP.1/KEYHMAC	A5074
RSA KeyGen/SigGen/SigVer – 2048/3072/4096	FIPS 186-4	FCS_CKM.1 FCS_COP.1/SIGN	A5074
SHS SHA-1/256/384/512	FIPS 180-4	FCS_COP.1/HASH	A5074
RSA Key Establishment	SP 800-56A	FCS_CKM.2/UNLOCKED	Tested with known good implementation
FFC Safe Primes	SP 800-56A	FCS_CKM.1 FCS_CKM.2/UNLOCKED	Tested with known good implementation

Table 4 - BoringSSL Cryptographic Algorithms

The Samsung Crypto Extension v1.0 library provides the following algorithms.

Algorithm	NIST Standard	SFR Reference	Cert#
KBKDF	SP 800-108	FCS_CKM_EXT.3	A933

Table 5 - Samsung Crypto Extension Cryptographic Algorithms

The evaluated devices utilize the following kernels for the Samsung Kernel Cryptographic Module (Kernel Crypto).

Device	Kernel Version	Kernel Crypto Version
Z Flip 6/Z Fold 6	6.1	2.4
S24 FE	6.1	2.4
A52 5G	4.19	2.2
A53 5G	5.10	2.2

Table 6 - Kernel Versions

The Samsung Kernel Cryptographic (“Kernel Crypto”) Module provides the following algorithms. Note that with and without PAA is equivalent to with and without crypto extensions.

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128/256 CBC	FIPS 197, SP 800-38A	FCS_COP.1/ENCRYPT	A5143
			A1456, A1455, A970, A969
HMAC SHA-1/256	FIPS 198-1 & 180-4	FCS_COP.1/KEYHMAC	A5143
			A1456, A1455, A970, A969
DRBG SHA-256 HMAC_DRBG	SP 800-90A	FCS_RBG_EXT.1	A5143
			A1456, A1455, A970, A969
SHS SHA-1/256	FIPS 180-4	FCS_COP.1/HASH	A5143
			A1456, A1455, A970, A969

Table 7 - Samsung Kernel Cryptographic Algorithms

The evaluated devices utilize the Samsung SCrypto Cryptographic Module for cryptographic operations within the TEE on each device. The following table lists the TEE operating systems for each device.

Device	SoC	TEE OS Version	SCrypto Version
Galaxy Z Flip6 5G	Snapdragon 8 Gen 3	QSEE 6.1	2.7
Galaxy S24 FE	Exynos 2400	TEEGRIS 5.0.0	2.7
Galaxy A52 5G	Snapdragon 750	QSEE 5.10	2.5
Galaxy A53 5G	Exynos 1280	TEEGRIS 4.2.1	2.6

Table 8 - TEE Environments

The Samsung SCrypto TEE library provides the following algorithms. Note that the TOE only performs RSA signing/decryption (using the private key) in the TEE, and performs public key verification/encryption in the normal world using BoringSSL.

Algorithm	NIST Standard	SFR Reference	Cert#
AES CBC/GCM 128/256	FIPS 197, SP 800-38A/D	FCS_COP.1/ENCRYPT	A3243 A915 A889
DRBG AES-256 CTR_DRBG	SP 800-90A	FCS_RBG_EXT.1	A3243 A915 A889
ECDSA KeyGen/KeyVer/SigGen/SigVer P-256/384/521	FIPS 186-4	FCS_CKM.1 FCS_COP.1/SIGN	A3243 A915 A889
HMAC SHA-1/256/384/512	FIPS 198-1 & 180-4	FCS_COP.1/KEYHMAC	A3243 A915 A889
RSA KeyGen and SigGen (no verification) 2048 bits	FIPS 186-4	FCS_CKM.1 FCS_CKM.2/UNLOCKED FCS_COP.1/SIGN	A3243 A915 A889
SHS SHA-1/256/384/512	FIPS 180-4	FCS_COP.1/HASH	A3243 A915 A889
KBKDF	SP 800-108	FCS_CKM_EXT.3	A3243 A915 A889

Table 9 - SCrypto TEE Cryptographic Algorithms

The Chipset hardware for storage encryption has various modules that provide cryptographic functions. The modules and versions are listed here. Only discrete modules are listed here.

Device	Flash Crypto
All ZFlip6(Qualcomm)	Qualcomm ICE v4.0.1
All S24x(Samsung)	Samsung sf_crypt_config_fmp_fx_sk_v3.00
A52 5G(Qualcomm)	Qualcomm ICE v3.1.0
A53 5G(Samsung)	Samsung FMP v4.0.1

Table 10 - Hardware Components

The storage encryption modules provide the following algorithms.

Algorithm	NIST Standard	SFR Reference	Cert#
XTS-AES 256 (Exynos FMP)	FIPS 197, SP 800-38E	FCS_COP.1/ENCRYPT	A5200, A2723
XTS-AES 128/256 (Qualcomm)	FIPS 197, SP 800-38E	FCS_COP.1/ENCRYPT	A4287, A4288 A1010, A1009

Table 11 - Storage Hardware Algorithms

The devices contain unique Wi-Fi chipsets based on the model of the device. The chipsets are listed here.

Device	Wi-Fi Chipset	Wi-Fi Chipset Details
Galaxy ZFlip Galaxy S24 FE	Qualcomm WCN7851	Incorporates Qualcomm's Qualcomm Lithium AES engine-256w core to provide WPA2/WPA3 AES encryption
A52 5G/A42 5G	Qualcomm WCN3988	WCN39xx are families of chips that support 802.11 WIFI technologies for Qualcomm Chipsets, including(but not limited to) SDM8xx (835 or later chipsets), SDM6xx/4xx chipsets. IPQ80XX and QCA6XXX are families of chips that support 802.11 WIFI technologies
A53 5G	Samsung S5N5C20X00-6030	The Samsung WLAN Cryptographic Accelerator is a hardware module that provides AES support for WLAN packet data in multiple Exynos products.

Table 12 - Wi-Fi Hardware Components

The Wi-Fi chipset hardware provides the following algorithms.

Algorithm	NIST Standard	SFR Reference	Cert#
AES 128 CCM (Qualcomm Wi-Fi)	FIPS 197, SP 800-38C	FCS_COP.1/ENCRYPT	A3238, 4748
AES 128 CCM (Samsung Wi-Fi)	FIPS 197, SP 800-38C	FCS_COP.1/ENCRYPT	C1623

Table 13 - Wi-Fi Chip Algorithms

Several devices provide a secure processor for mutable hardware key storage. The devices and chipsets are listed here.

Device	Mutable Storage Chipset
All S24FE(Samsung)	Smart Secure Platform 7.0
All ZFlip6 (Qualcomm)	Secure Processing Unit 1.3

Table 14 - Mutable Key Storage Components

The chipsets for the mutable hardware key storage provide the following algorithms.

Algorithm	NIST Standard	SFR Reference	Cert#
AES CBC/GCM 128/256	FIPS 197, SP 800-38A/D	FCS_COP.1/ENCRYPT	A4780, A4779, A4445, A4444
DRBG Hash_DRBG (Qualcomm)	SP 800-90A	FCS_RBG_EXT.1	A2840
ECDSA SigGen/SigVer (P-256/384)	FIPS 186-4	FCS_COP.1/SIGN	A4780, A4445

Algorithm	NIST Standard	SFR Reference	Cert#
HMAC SHA-256/384/512 (Qualcomm) HMAC SHA-256 (Exynos)	FIPS 198-1 & 180-4	FCS_COP.1/KEYHMAC	A4780, A4444
RSA SigGen/SigVer - 2048	FIPS 186-4	FCS_COP.1/SIGN	A4780, A4445
SHS SHA-256	FIPS 180-4	FCS_COP.1/HASH	A4779, A4444

Table 15 - Mutable Key Storage Cryptographic Algorithms

The SoC hardware provides the following algorithms.

Algorithm	NIST Standard	SFR Reference	Cert#
KBKDF (Exynos)	SP 800-108	FCS_CKM_EXT.3	A5161
			A2536
AES 128/256 CBC/GCM (Exynos)	FIPS 197, SP 800-38A/D	FCS_COP.1/ENCRYPT	A5161
			A2536
SHS SHA-256 (Exynos)	FIPS 180-4	FCS_COP.1/HASH	A5161
			A2536
HMAC SHA-256 (Exynos)	FIPS 198-1 & 180-4	FCS_COP.1/KEYHMAC	A5161
			A2536
AES 128/256 CBC (Qualcomm)	FIPS 197, SP 800-38A	FCS_COP.1/ENCRYPT	A4289
			A242
DRBG SHA-256 Hash_DRBG (Qualcomm)	SP 800-90A	FCS_RBG_EXT.1	A4298
			A50
SHS SHA-256 (Qualcomm)	FIPS 180-4	FCS_COP.1/HASH	A4289
			A242

Table 16 - SoC Cryptographic Algorithms

2. Protection Profile SFR Assurance Activities

This section of the AAR identifies each of the assurance activities included in the claimed Protection Profile and describes the findings in each case.

The following evidence was used to complete the Assurance Activities:

AAR v0.1

- Samsung Electronics Co., Ltd. Samsung Galaxy Devices on Android 14 – Fall Security Target, Version 0.3, 12/16/2024 **[ST]**
- Samsung Android 14 on Galaxy Devices Administrator Guide, version 9.0.3, December 16, 2024 **[Admin Guide]**

2.1 Security audit (FAU)

2.1.1 Audit Data Generation (MDFPP33:FAU_GEN.1)

2.1.1.1 MDFPP33:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 MDFPP33:FAU_GEN.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the PP is described and that the description of the fields contains the information required in FAU_GEN.1.2.

Section 6.1 of the ST provides a table of audit events. This table and surrounding text incudes all required events, the contents of each audit record, and which audit log stores the event. This table matches what is required in MDFPP33:FAU_GEN.1.2

Component Guidance Assurance Activities: The evaluator shall also make a determination of the administrative actions that are relevant in the context of this PP including those listed in the Management section. The evaluator shall examine the administrative guide and make a determination of which administrative commands are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the PP. The evaluator shall document the methodology or approach taken while determining which actions in the administrative guide are security relevant with respect to this PP. The evaluator may perform this activity as part of the activities associated with ensuring the AGD_OPE guidance satisfies the requirements.

Section 5.2 of the Admin Guide provides a spreadsheet which list the audit events corresponding with the requirements claimed in the ST and with the associated audit events for those requirements from the MDFPP. The evaluator confirmed the mapping is complete. Each of the listed events provides the required contents matching up to the MDFPP33:FAU_GEN.1 requirement. Section 5.1 of the Admin Guide contains the details of each audit record.

As part of testing, the evaluator verified each record in the Admin Guide and provided a sample of each.

Component Testing Assurance Activities: The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the provided table and administrative actions. This should include all instances of an event. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. For administrative actions, the evaluator shall test that each action determined by the evaluator above to be security relevant in the context of this PP is auditable. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields specified in FAU_GEN.1.2 are contained in each audit record.

Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly. For example, testing performed to ensure that the administrative guidance provided is correct verifies that AGD_OPE.1 is satisfied and should address the invocation of the administrative actions that are needed to verify the audit records are generated as expected.

The evaluator tested the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the provided tables above including all administrative actions. Section 5.2 of the Admin Guide includes audit events for the TOE and administrative actions. The evaluator collected these audit records while running the security functional tests.   When verifying the test results, the evaluator verified that the audit records generated during testing matched the format specified in the administrative guide, and that the fields in each audit record have the proper entries. For each type of audit record, the evaluator found that the TOE correctly generated an audit log matching the vendor specified one. The evaluator collected a sample of each type of audit record and included these samples in the Detailed Test Report for this evaluation.

2.1.2 Audit Data Generation (Bluetooth) - per TD0707 (BT10:FAU_GEN.1/BT)

2.1.2.1 BT10:FAU_GEN.1.1/BT

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.2.2 BT10:FAU_GEN.1.2/BT

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.1.3 Audit Data Generation (Wireless LAN) (WLANC10:FAU_GEN.1/WLAN)

2.1.3.1 WLANC10:FAU_GEN.1.1/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.2 WLANC10:FAU_GEN.1.2/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS and ensure it provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field.

If 'invoke platform-provided functionality' is selected, the evaluator shall examine the TSS to verify it describes (for each supported platform) how this functionality is invoked (it should be noted that this may be through a mechanism that is not implemented by the WLAN Client; however, that mechanism will be identified in the TSS as part of this evaluation activity).

Section 6.1 of the ST enumerates the audit records and provides a reference to Section 5.1 in the Admin Guide for the format of the audit records.

Component Guidance Assurance Activities: The evaluator shall check the operational guidance and ensure it lists all of the auditable events and provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the PP-Module is described and that the description of the fields contains the information required in FAU_GEN.1.2/WLAN, and the additional information specified in Table 2 in the main document and Table 5 in the main document.

The evaluator shall in particular ensure that the operational guidance is clear in relation to the contents for failed cryptographic events. In the Auditable Events tables, information detailing the cryptographic mode of operation and a name or identifier for the object being encrypted is required. The evaluator shall ensure that name or identifier is sufficient to allow an administrator reviewing the audit log to determine the context of the cryptographic operation (for example, performed during a key negotiation exchange, performed when encrypting data for transit) as well as the non-TOE endpoint of the connection for cryptographic failures relating to communications with other IT systems.

The evaluator shall also make a determination of the administrative actions that are relevant in the context of this PP-Module. The TOE may contain functionality that is not evaluated in the context of this PP-Module because the functionality is not specified in an SFR. This functionality may have administrative aspects that are described in the operational guidance. Since such administrative actions will not be performed in an evaluated configuration of the TOE, the evaluator shall examine the operational guidance and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the PP-Module, which thus form the set of 'all administrative actions'. The evaluator may perform this activity as part of the activities associated with ensuring the AGD_OPE guidance satisfies the requirements.

See MDFPP33:FAU_GEN.1

Component Testing Assurance Activities: The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records in accordance with the assurance activities associated with the functional requirements in this PP-Module. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.

Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly. For example, testing performed to ensure that the administrative guidance provided is correct verifies that AGD_OPE.1 is satisfied and should address the invocation of the administrative actions that are needed to verify the audit records are generated as expected.

See MDFPP33:FAU_GEN.1

2.1.4 Audit Review (MDFPP33:FAU_SAR.1)

2.1.4.1 MDFPP33:FAU_SAR.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.2 MDFPP33:FAU_SAR.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluation activity for this requirement is performed in conjunction with test for function 32 of FMT_SMF.1.

See function 32 of MDFPP33:FMT_SMF_EXT.1 for the audit review test

2.1.5 Audit Storage Protection (MDFPP33:FAU_STG.1)

2.1.5.1 MDFPP33:FAU_STG.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.5.2 MDFPP33:FAU_STG.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS lists the location of all logs and the access controls of those files such that unauthorized modification and deletion are prevented.

Section 6.1 of the ST states that the TOE stores audit records in a file within the file system accessible only to Linux processes with system permissions (effectively the TSF itself and MDM agents using the defined APIs). These restrictions prevent the unauthorized modification or deletion of the audit records stored in the audit files. The exact location is provided in the KMD. These restrictions prevent the unauthorized modification or deletion of the audit records stored in the audit files.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Test 1: The evaluator shall attempt to delete the audit trail in a manner that the access controls should prevent (as an unauthorized user) and shall verify that the attempt fails.

Test 2: The evaluator shall attempt to modify the audit trail in a manner that the access controls should prevent (as an unauthorized application) and shall verify that the attempt fails.

The TOE protects the security log in memory, and only allows access to the logd daemon, which only affords a device owner the MDM API to retrieve a copy of these logs. Because of this, the evaluator had no method to delete or modify the logs present in memory. Only an authorized administrator can read the audit trail via the TOE's MDM APIs.

Test 1: The evaluator attempted to delete the audit trail as an unauthorized user and confirmed that there were no access controls of any kind to access or delete the logs.

Test 2: The evaluator tested both modification and removal and found no way to attempt modification or removal of the admin protected audit logs.

2.1.6 Prevention of Audit Data Loss (MDFPP33:FAU_STG.4)

2.1.6.1 MDFPP33:FAU_STG.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it describes the size limits on the audit records, the detection of a full audit trail, and the actions taken by the TSF when the audit trail is full. The evaluator shall ensure that the actions results in the deletion or overwrite of the oldest stored record.

Section 6.1 of the ST states that the TOE pre-allocates a file system area (between 10MB and 50MB in size, depending upon available storage on the device) by creating a /data/system/[admin_uid]_bubble/bubbleFile and directory (/data/system/[admin_uid]) in which to archive compressed audit logs. If the TOE lacks sufficient space (at least 10MB), then the TOE returns a failure code in response to the administrator's attempt to enable the AuditLog. Once enabled, the TOE writes audit events into nodes until they read a given size, and then compresses and archives the records. The TOE utilizes a circular buffer approach to handle when the accumulated, compressed audit events exceed the allocated file system size. When the limit is reached, the TOE removes the oldest audit logs, freeing space for new records.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2 Cryptographic support (FCS)

2.2.1 Cryptographic Key Generation (MDFPP33:FCS_CKM.1)

2.2.1.1 MDFPP33:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

The table in the FCS_CKM.1 section of the ST specifies which cryptographic libraries support RSA, DH, ECDH, and ECDSA. The TOE generates RSA keys in its SCrypto library and generates DH/ECDH/ECDSA (including P-256, P384 and P-521) keys in BoringSSL and ECDSA (including P-256, P384 and P-521) keys in in SCrypto. The TOE supports generating keys with a security strength of 112-bits and larger, thus supports 2048-bit RSA and DH keys, and 256-bit ECDH/ECDSA keys. The usage for each scheme is described in MDFPP33:FCS_CKM.2.

Cryptographic Library	RSA Generation	DH (FFC)	ECDH (ECC)	EDCSA (ECC)
BoringSSL (user space)	Yes	Yes	Yes	Yes
Kernel Crypto (Kernel)	No	No	No	No
SCrypto (TrustZone)	Yes	No	No	Yes
Application Processor	No	No	No	No

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation schemes and key sizes for all uses defined in this PP.

The Admin Guide, section 3.1, explains that when the TOE is in CC mode, it will only use approved cryptographic functions. No additional configuration is needed beyond putting the device in CC mode.

Component Testing Assurance Activities: Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d.

Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

1. Random Primes:

o Provable primes

o Probable primes

2. Primes with Conditions:

o Primes p1, p2, q1,q2, p and q shall all be provable primes

o Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes

o Primes p1, p2, q1,q2, p and q shall all be probable primes

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seeds, the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

If possible, the Random Probable primes method should also be verified against a known good implementation as described above. Otherwise, the evaluator shall have the TSF generate 10 keys pairs for each supported key length nlen and verify:

- n = p*q

- p and q are probably prime according to Miller-Rabin tests

- GCD(p-1,e) = 1

- GCD(q-1,e) = 1

- 2^16 < e < 2^256 and e is an odd integer

- |p-q| > 2^(nlen/2 â€“ 100)

- p >= squareroot(2)*(2^(nlen/2 -1))

- q >= squareroot(2)*(2^(nlen/2 -1))

- 2^(nlen/2) < d < LCM(p-1,q-1)

- e*d = 1 mod LCM(p-1,q-1)

Key Generation for FIPS 186-4 Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

For each supported NIST curve, i.e. P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test

For each supported NIST curve, i.e. P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e. correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Curve25519

The evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated as specified in RFC 7748 using an approved random bit generator (RBG) and shall be written in little-endian order (least significant byte first). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

Note: Assuming the PKV function of the good implementation will (using little-endian order):

a. Confirm the private and public keys are 32-byte values

b. Confirm the three least significant bits of the first byte of the private key are zero

c. Confirm the most significant bit of the last byte is zero

d. Confirm the second most significant bit of the last byte is one

e. Calculate the expected public key from the private key and confirm it matches the supplied public key

The evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify 5 of the public key values so that they are incorrect, leaving five values unchanged (i.e. correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.

The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

Cryptographic and Field Primes:

- Primes q and p shall both be provable primes
- Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

Cryptographic Group Generator:

- Generator g constructed through a verifiable process
- Generator g constructed through an unverifiable process

The Key generation specifies 2 ways to generate the private key x:

Private Key:

- len(q) bit output of RBG where 1 <= x <= q-1
- len(q) + 64 bit output of RBG, followed by a mod q-1 operation where 1<= x<=q-1

The security strength of the RBG must be at least that of the security offered by the FFC parameter set.

To test the cryptographic and field prime generation method for the provable primes method or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.

For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- g != 0,1
- q divides p-1
- g^q mod p = 1
- g^x mod p = y

for each FFC parameter set and key pair.

See Section 1.2 for a listing of applicable CAVP certificates.

2.2.2 Cryptographic Key Generation (VPNC24:FCS_CKM.1)

2.2.2.1 VPNC24:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

See the description for MDFPP33:FCS_CKM.1 above.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all uses defined in this PP.

The Admin Guide, section 3.1, explains that when the TOE is in CC mode, it will only use approved cryptographic functions. No additional configuration is needed beyond putting the device in CC mode.

Component Testing Assurance Activities: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d.

Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

1. Random Primes:

Provable primes

Probable primes

2. Primes with Conditions:

Primes p1, p2, q1, q2, p and q shall all be provable primes

Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes

Primes p1, p2, q1, q2, p and q shall all be probable primes

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

If possible, the Random Probable primes method should also be verified against a known good implementation as described above. Otherwise, the evaluator shall have the TSF generate 10 keys pairs for each supported key length nlen and verify:

n = p*q

p and q are probably prime according to Miller-Rabin tests

GCD(p-1,e) = 1

GCD(q-1,e) = 1

2^16 < e < 2^256 and e is an odd integer

|p-q| > 2^(nlen/2 - 100)

p >= squareroot(2)*(2^(nlen/2 -1))

q >= squareroot(2)*(2^(nlen/2 -1))

2^(nlen/2) < d < LCM(p-1,q-1)

e*d = 1 mod LCM(p-1,q-1)

Key Generation for FIPS 186-4 Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Curve25519

The evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated as specified in RFC 7748 using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

Note: Assuming the PKV function of the good implementation will:

a. confirm the private and public keys are 32-byte values

b. confirm the three least significant bits of the most significant byte of the private key are zero

c. confirm the most significant bit of the least significant byte is zero

d. confirm the second most significant bit of the most significant byte is one

e. calculate the expected public key from the private key and confirm it matches the supplied public key

The evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify 5 of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.

The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

Cryptographic and Field Primes:

Primes q and p shall both be provable primes

Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

Cryptographic Group Generator:

Generator g constructed through a verifiable process

Generator g constructed through an unverifiable process

The Key generation specifies 2 ways to generate the private key x:

Private Key:

len(q) bit output of RBG where 1 <= x <= q-1

len(q) + 64 bit output of RBG, followed by a mod q-1 operation where 1<= x<=q-1

The security strength of the RBG must be at least that of the security offered by the FFC parameter set.

To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.

For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

Verification must also confirm

g != 0,1

q divides p-1

g^q mod p = 1

g^x mod p = y

for each FFC parameter set and key pair.

Testing for FFC Schemes using Diffie-Hellman group 14 is done as part of testing in CKM.2.1.

See Section 1.2 for a listing of applicable CAVP certificates.

2.2.3 VPN Cryptographic Key Generation (IKE) (VPNC24:FCS_CKM.1/VPN)

2.2.3.1 VPNC24:FCS_CKM.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes how the key generation functionality is invoked.

See MDFPP33:FCS_CKM.1 where the cryptographic libraries are identified for invocation.

Component Guidance Assurance Activities: None Defined.

Component Testing Assurance Activities: If this functionality is implemented by the TSF, refer to the following EAs, depending on the TOE's claimed Base-PP:

- GPOS PP: FCS_CKM.1

- MDF PP: FCS_CKM.1

- App PP: FCS_CKM.1/AK

- MDM PP: FCS_CKM.1

See MDFPP33:FCS_CKM.1

2.2.4 Cryptographic Key Generation (Symmetric Keys for WPA2/WPA3 Connections) (WLANC10:FCS_CKM.1/WPA)

2.2.4.1 WLANC10:FCS_CKM.1.1/WPA

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes how the primitives defined and implemented by this PP-Module are used by the TOE in establishing and maintaining secure connectivity to the wireless clients. The TSS shall also provide a description of the developer's method(s) of assuring that their implementation conforms to the cryptographic standards; this includes not only testing done by the developing organization, but also any third-party testing that is performed.

Section 6.2 of the ST explains the TOE adheres to IEEE 802.11-2012 and IEEE 802.11ac-2014 for key generation. The TOE’s wpa_supplicant provides the PRF384 and PRF704 for WPA3/WPA2 derivation of 128-bit or 256-bit AES Temporal Key (using the HMAC implementation provided by BoringSSL) and employs its BoringSSL AES-256 DRBG when generating random values used in the EAP-TLS and 802.11 4-way handshake. The TOE supports the AES-128 CCMP encryption mode. The TOE has successfully completed certification (including WPA3/WPA2 Enterprise) and received Wi-Fi CERTIFIED Interoperability Certificates from the Wi-Fi Alliance. The Wi-Fi Alliance maintains a website providing further information about the testing program: http://www.wi-fi.org/certification.

Device Name	Model Number	Wi-Fi Alliance Certificate Numbers
Galaxy A52	SM-G78xx	110242, 110243, 110475, 110613, 110614
Galaxy Z Fold6 5G	SM-F956x	125692, 126097, 126098, 126099, 126164, 126165
Galaxy Z Flip6 5G	SM-F741x	125544, 126061, 126063, 126223, 126224
Galaxy Z Fold SE	SM-F958	131717
Galaxy A53 5G	SM-A536 SM-S536	117220, 117304, 117536, 117578, 117579, 117580, 117581, 117880, 117881, 117882, 117902, 1211138
Galaxy S24 FE	SM-S721x	131705, 132146, 132147, 132247, 132248

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall configure the access point so the cryptoperiod of the session key is 1 hour. The evaluator shall successfully connect the TOE to the access point and maintain the connection for a length of time that is greater than the configured cryptoperiod. The evaluator shall use a packet capture tool to determine that after the configured cryptoperiod, a re-negotiation is initiated to establish a new session key. Finally, the evaluator shall determine that the renegotiation has been successful and the client continues communication with the access point.

Test 2: The evaluator shall perform the following test using a packet sniffing tool to collect frames between the TOE and a wireless LAN access point:

Step 1: The evaluator shall configure the access point to an unused channel and configure the WLAN sniffer to sniff only on that channel (i.e., lock the sniffer on the selected channel). The sniffer should also be configured to filter on the MAC address of the TOE and/or access point.

Step 2: The evaluator shall configure the TOE to communicate with a WLAN access point using IEEE 802.11-2012 and a 256-bit (64 hex values 0-f) pre-shared key. The pre-shared key is only used for testing.

Step 3: The evaluator shall start the sniffing tool, initiate a connection between the TOE and the access point, and allow the TOE to authenticate, associate, and successfully complete the 4-way handshake with the client.

Step 4: The evaluator shall set a timer for 1 minute, at the end of which the evaluator shall disconnect the TOE from the wireless network and stop the sniffer.

Step 5: The evaluator shall identify the 4-way handshake frames (denoted EAPOL-key in Wireshark captures) and derive the PTK from the 4-way handshake frames and pre-shared key as specified in IEEE 802.11-2012.

Step 6: The evaluator shall select the first data frame from the captured packets that was sent between the TOE and access point after the 4-way handshake successfully completed, and without the frame control value 0x4208 (the first 2 bytes are 08 42). The evaluator shall use the PTK to decrypt the data portion of the packet as specified in IEEE 802.11-2012, and shall verify that the decrypted data contains ASCII-readable text.

Step 7: The evaluator shall repeat Step 6 for the next 2 data frames between the TOE and access point and without frame control value 0x4208.

frame control value 0x4208.

Test 1 – The access point was configured for a 1-hour cryptoperiod. The TOE was configured to connect to the access point and a wireless packet capture was started. After an hour, the evaluator examined the packet capture and ensured there was a re-key between the access point and the TOE.

Test 2 - The TOE was configured to connect to an access point and a wireless packet capture was started. The TOE was connected and disconnected after a minute once a number of broadcast packets were observed using the packet capture tool. The evaluator filtered the capture further to demonstrate the 4-way handshake and encrypted broadcast packets. The evaluator then decrypted the packet capture and demonstrated the PTK and GTK were derived.

See Section 1.2 for identification of CAVP certificates that map to this requirement (AES and HMAC).

2.2.5 Cryptographic Key Establishment (MDFPP33:FCS_CKM.2/LOCKED)

2.2.5.1 MDFPP33:FCS_CKM.2.1/LOCKED

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The test for SP800-56A and SP800-56B key establishment schemes is performed in association with FCS_CKM.2/UNLOCKED.

Curve25519 Key Establishment Schemes

The evaluator shall verify a TOE's implementation of the key agreement scheme using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specification. These components include the calculation of the shared secret K and the hash of K.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement role and hash function combination, the tester shall generate 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the shared secret value K, and the hash of K.

The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value K and compare the hash generated from this value.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results. To conduct this test, the evaluator generates a set of 30 test vectors consisting of data sets including the evaluator's public keys and the TOE's public/private key pairs.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value K or the hash of K. At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

See Section 1.2 for a listing of applicable CAVP certificates.

2.2.6 Cryptographic Key Establishment (VPNC24:FCS_CKM.2/UNLOCK)

2.2.6.1 VPNC24:FCS_CKM.2.1/UNLOCK

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

If 'Key establishment scheme using Diffie-Hellman group 14' is selected, the evaluator shall ensure that the TSS describes how the implementation meets RFC 3526 Section 3.

See MDFPP33:FCS_CKM.2/UNLOCK

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

The Admin Guide, section 3.1, explains that when the TOE is in CC mode, it will only use approved cryptographic functions. No additional configuration is needed beyond putting the device in CC mode.

Component Testing Assurance Activities: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Key Establishment Schemes

The evaluator shall verify the implementation of the key establishment schemes supported by the TOE using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding TOE's public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information field OI and TOE id fields.

If the TOE does not use a KDF defined in SP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

SP800-56B Key Establishment Schemes

The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

If the TOE acts as a sender, the following assurance activity shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

If the TOE acts as a receiver, the following assurance activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM-KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

If 'Key establishment scheme using Diffie-Hellman group 14 is selected, the evaluator shall also verify the correctness of the TSF's implementation of Diffie-Hellman group 14 by using a known good implementation for each protocol selected in FTP_ITC_EXT.1 in the MDF PP that uses Diffie-Hellman group 14. Note that because a TOE that conforms to this PP-Module must implement IPsec, the tested protocols shall include IPsec at minimum.

See Section 1.2 for a listing of applicable CAVP certificates.

The evaluator tested with a known good implementation to verify the DH-14 implementation.

2.2.7 Cryptographic Key Establishment (MDFPP33:FCS_CKM.2/UNLOCKED)

2.2.7.1 MDFPP33:FCS_CKM.2.1/UNLOCKED

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

Section 6.2 of the ST states that the TOE supports RSA (800-56B, as an initiator only), DHE (FFC 800-56A), and ECDHE (ECC 800-56A) methods in TLS key establishment/exchange. The TOE has CVL KAS and ECDSA CAVP algorithm certificates for Elliptic Curve key establishment and key generation respectively as identified in Section 1.2. Samsung vendor-affirms that the TOE's RSA key establishment follows 800-56B. The TOE implementation meets RFC 3526 Section 3. The user and administrator need take no special configuration of the TOE as the TOE automatically generates the keys needed for negotiated TLS ciphersuites. Because the TOE only acts as a TLS client, the TOE only performs 800-56B encryption (specifically the encryption of the Pre-Master Secret using the Server's RSA public key) when participating in TLS_RSA_* based TLS handshakes. Thus, the TOE does not perform 800-56B decryption. However, the TOE's TLS client correctly handles other cryptographic errors (for example, invalid checksums, incorrect certificate types, corrupted certificates) by sending a TLS fatal alert.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment schemes.

The Admin Guide, section 3.1, explains that when the TOE is in CC mode, it will only use approved cryptographic functions. No additional configuration is needed beyond putting the device in CC mode.

Component Testing Assurance Activities: Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

The evaluator shall verify the implementation of the key establishment schemes supported by the TOE using the applicable tests below.

SP800-56A Revision 3 Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A Revision 3 key establishment schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MacTag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding TOE's public keys (static or ephemeral), the MAC tags, and any inputs used in the KDF, such as the Other Information field OI and TOE id fields.

If the TOE does not use a KDF defined in SP 800-56A Revision 3, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A Revision 3 key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MacTag, and any inputs used in the KDF, such as the other info and TOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MacTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

SP800-56B Key Establishment Schemes

The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

If the TOE acts as a sender, the following evaluation activity shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme: To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

If the TOE acts as a receiver, the following evaluation activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme: To conduct this test the evaluator shall generate or obtain test vectors FCS_CKM.2.1/LOCKED from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEMKWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

RSAES-PKCS1-v1_5 Key Establishment Schemes

The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_ITC_EXT.1 that uses RSAES-PKCS1-v1_5.

FFC Schemes using 'safe-prime' groups

The evaluator shall verify the correctness of the TSF's implementation of 'safe-prime' groups by using a known good implementation for each protocol selected in FTP_ITC_EXT.1 that uses 'safe-prime' groups. This test must be performed for each 'safe-prime' group that each protocol uses.

See Section 1.2 for a listing of applicable CAVP certificates.

The evaluator tested with a known good implementation to verify the DH-14 implementation. The TOE's RSA key exchange mechanism is used in the TLS handshake process and during both product development and evaluation testing, the TOE's implementation undergoes testing to ensure TLS compatibility. Any defect in the RSA key exchange mechanism would result in an inability to negotiate TLS_RSA_* ciphersuites with a separate, known good implementation. Gossamer's TLSC testing results demonstrate the TOE conforms to the specifications.

2.2.8 Cryptographic Key Distribution (Group Temporal Key for WLAN) (WLANC10:FCS_CKM.2/WLAN)

2.2.8.1 WLANC10:FCS_CKM.2.1/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes how the GTK is unwrapped prior to being installed for use on the TOE using the AES implementation specified in this PP-Module.

Section 6.2 of the ST states the TOE adheres to RFC 3394, SP 800-38F, and 802.11-2012 standards and unwraps the GTK (sent encrypted with the WPA3/2 KEK using AES Key Wrap in an EAPOL-Key frame). The TOE, upon receiving an EAPOL frame, will subject the frame to a number of checks (frame length, EAPOL version, frame payload size, EAPOL-Key type, key data length, EAPOL-Key CCMP descriptor version, and replay counter) to ensure a proper

EAPOL message and then decrypt the GTK using the KEK, thus ensuring that it does not expose the Group Temporal Key (GTK).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following test using a packet sniffing tool to collect frames between the TOE and a wireless access point (which may be performed in conjunction with the assurance activity for FCS_CKM.1.1/WLAN).

Step 1: The evaluator shall configure the access point to an unused channel and configure the WLAN sniffer to sniff only on that channel (i.e., lock the sniffer on the selected channel). The sniffer should also be configured to filter on the MAC address of the TOE and/or access point.

Step 2: The evaluator shall configure the TOE to communicate with the access point using IEEE 802.11-2012 and a 256-bit (64 hex values 0-f) pre-shared key, setting up the connections as described in the operational guidance. The pre-shared key is only used for testing.

Step 3: The evaluator shall start the sniffing tool, initiate a connection between the TOE and access point, and allow the TOE to authenticate, associate, and successfully complete the 4-way handshake with the TOE.

Step 4: The evaluator shall set a timer for 1 minute, at the end of which the evaluator shall disconnect the TOE from the access point and stop the sniffer.

Step 5: The evaluator shall identify the 4-way handshake frames (denoted EAPOL-key in Wireshark captures) and derive the PTK and GTK from the 4-way handshake frames and pre-shared key as specified in IEEE 802.11-2012.

Step 6: The evaluator shall select the first data frame from the captured packets that was sent between the TOE and access point after the 4-way handshake successfully completed, and with the frame control value 0x4208 (the first 2 bytes are 08 42). The evaluator shall use the GTK to decrypt the data portion of the selected packet as specified in IEEE 802.11-2012, and shall verify that the decrypted data contains ASCIIreadable text.

Step 7: The evaluator shall repeat Step 6 for the next 2 data frames with frame control value 0x4208.

See the test case for WLANC10:FCS_CKM.1/WPA.

See Section 1.2 for identification of CAVP certificates that map to this requirement

2.2.9 Cryptographic Key Support (MDFPP33:FCS_CKM_EXT.1)

2.2.9.1 MDFPP33:FCS_CKM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.9.2 MDFPP33:FCS_CKM_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.9.3 MDFPP33:FCS_CKM_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall review the TSS to determine that a REK is supported by the TOE, that the TSS includes a description of the protection provided by the TOE for a REK, and that the TSS includes a description of the method of generation of a REK.

The evaluator shall verify that the description of the protection of a REK describes how any reading, import, and export of that REK is prevented. For example, if the hardware protecting the REK is removable, the description should include how other devices are prevented from reading the REK. The evaluator shall verify that the TSS describes how encryption/decryption/derivation actions are isolated so as to prevent applications and systemlevel processes from reading the REK while allowing encryption/decryption/derivation by the key.

The evaluator shall verify that the description includes how the OS is prevented from accessing the memory containing REK key material, which software is allowed access to the REK, how any other software in the execution environment is prevented from reading that key material, and what other mechanisms prevent the REK key material from being written to shared memory locations between the OS and the separate execution environment.

If key derivation is performed using a REK, the evaluator shall ensure that the TSS description includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to FCS_CKM_EXT.3.2.

The evaluator shall verify that the generation of a REK meets the FCS_RBG_EXT.1.1 and FCS_RBG_EXT.1.2 requirements:

- If REKs is/are generated on-device, the TSS shall include a description of the generation mechanism including what triggers a generation, how the functionality described by FCS_RBG_EXT.1 is invoked, and whether a separate instance of the RBG is used for REKs.

- If REKs is/are generated off-device, the TSS shall include evidence that the RBG meets FCS_RBG_EXT.1. This will likely necessitate a second set of RBG documentation equivalent to the documentation provided for the RBG Evaluation Activities. In addition, the TSS shall describe the manufacturing process that prevents the device manufacturer from accessing any REKs.

Section 6.2 of the TSS states the TOE supports a Root Encryption Key (REK) within the main (application) processor. Requests for encryption or decryption chaining to the REK are only accessible through the Trusted Execution Environment, or TEE (TrustZone). The REK lies in a series of 256-bit fuses, programmed during manufacturing. The TEE does not allow direct access to the REK but provides services to derive a HEK (Hardware Encryption Key, which is derived from the REK through a KDF function) for encryption and decryption.

The REK value is generated during manufacturing either by the TOE (if it detects that the REK fuses have not been set) using its hardware DRBG or is generated during fabrication using an external RBG that meets the requirements of this PP in that the process utilizes a SHA-256 Hash_DRBG seeded by a hardware entropy source identical in architecture to that within the TOE. This fabrication process includes strict controls (including physical and logical access control to the manufacturing room where programming takes place as well as video surveillance and access only to specific, authorized, trusted individuals) to ensure that the fabricator cannot access any REK values between generation and programming.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.10 Cryptographic Key Random Generation (MDFPP33:FCS_CKM_EXT.2)

2.2.10.1 MDFPP33:FCS_CKM_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the documentation of the product's encryption key management is detailed enough that, after reading, the product's key management hierarchy is clear and that it meets the requirements to ensure the keys are adequately protected. The evaluator shall ensure that the documentation includes both an essay and one or more diagrams. Note that this may also be documented as separate proprietary evidence rather than being included in the TSS.

The evaluator shall also examine the key hierarchy section of the TSS to ensure that the formation of all DEKs is described and that the key sizes match that described by the ST author. The evaluator shall examine the key hierarchy section of the TSS to ensure that each DEK is generated or combined from keys of equal or greater security strength using one of the selected methods.

- If the symmetric DEK is generated by an RBG, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1 or documentation available for the operational environment to determine that the key size being requested is greater than or equal to the key size and mode to be used for the encryption/decryption of the data.

- If the DEK is formed from a combination, the evaluator shall verify that the TSS describes the method of combination and that this method is either an XOR or a KDF to justify that the effective entropy of each factor is preserved. The evaluator shall also verify that each combined value was originally generated from an Approved DRBG described in FCS_RBG_EXT.1.

- If concatenating the keys and using a KDF (as described in (SP 800-56C) is selected, the evaluator shall ensure the TSS includes a description of the randomness extraction step.

The description must include how an approved untruncated MAC function is being used for the randomness extraction step and the evaluator must verify the TSS describes that the output length (in bits) of the MAC function is at least as large as the targeted security strength (in bits) of the parameter set employed by the key establishment scheme (see Tables 1-3 of SP 800-56C).

The description must include how the MAC function being used for the randomness extraction step is related to the PRF used in the key expansion and verify the TSS description includes the correct MAC function:

- If an HMAC-hash is used in the randomness extraction step, then the same HMAC-hash (with the same hash function hash) is used as the PRF in the key expansion step.

- If an AES-CMAC (with key length 128, 192, or 256 bits) is used in the randomness extraction step, then AES-CMAC with a 128-bit key is used as the PRF in the key expansion step.

- The description must include the lengths of the salt values being used in the randomness extraction step and the evaluator shall verify the TSS description includes correct salt lengths:

- If an HMAC-hash is being used as the MAC, the salt length can be any value up to the maximum bit length permitted for input to the hash function hash.

- If an AES-CMAC is being used as the MAC, the salt length shall be the same length as the AES key (i.e. 128, 192, or 256 bits).

(conditional) If a KDF is used, the evaluator shall ensure that the TSS includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108 or SP 800-56C.

Section 6.2 of the TSS explains that the TOE supports Data Encryption Key (DEK) generation using its approved RBGs for use in SD card encryption. The TOE RBGs are capable of generating AES 256-bit DEKs in response to applications and services on the device. These can be accessed through both Android native APIs and C APIs depending on the library being called. For FBE, the TOE supports using a SP800-108 KDF to concatenate keys together to generate unique DEKs. The keys used in the SP800-108 KDF are generated by the approved RBGs. The TOE can also generate 128-bit asymmetric keys used for sensitive data protection. The proprietary key hierarchy diagrams in the KMD demonstrate all the required cryptographic operations. The evaluator examined the proprietary key hierarchy diagrams and verified that each DEK is generated or combined from keys of equal or greater security strength using one of the selected methods from the SFR.

Component Guidance Assurance Activities: The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1 or documentation available for the operational environment to determine that the key size being generated or combined is identical to the key size and mode to be used for the encryption/decryption of the data.

Section 6.2 of the TSS explains that the TOE supports Data Encryption Key (DEK) generation using its approved RBGs for use in SD card encryption. The TOE RBGs are capable of generating AES 256-bit DEKs in response to applications and services on the device. The 256-bit length matches the FCS_RBG_EXT.1 requirement.

Component Testing Assurance Activities: If a KDF is used, the evaluator shall perform one or more of the following tests to verify the correctness of the key derivation function, depending on the modes that are supported. Table 4 maps the data fields to the notations used in SP 800-108 and SP 800-56C.

Table 4: Notations used in SP 800-108 and SP 800-56C

Data Fields	Notations	
	SP 800-108	SP 800-56C
Pseudorandom function	PRF	PRF

Counter length	r	r
Length of output of PRF	h	h
Length of derived keying material	L	L
Length of input values	I length	I length
Pseudorandom input values I	K1 (key derivation key)	Z (shared secret)
Pseudorandom salt values	n/a	s
Randomness extraction MAC	n/a	MAC

Counter Mode Tests:

The evaluator shall determine the following characteristics of the key derivation function:

- One or more pseudorandom functions that are supported by the implementation (PRF).

- One or more of the values 8, 16, 24, 32 that equal the length of the binary representation of the counter (r).

- The length (in bits) of the output of the PRF (h).

- Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.

- Up to two values of L that are NOT evenly divisible by h.

- Location of the counter relative to fixed input data: before, after, or in the middle.

-- Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.

-- Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.

-- Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.

- The length (I_length) of the input values I.

For each supported combination of I_length, MAC, salt, PRF, counter location, value of r, and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I, and pseudorandom salt values. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it. For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

Feedback Mode Tests:

The evaluator shall determine the following characteristics of the key derivation function:

- One or more pseudorandom functions that are supported by the implementation (PRF).

- The length (in bits) of the output of the PRF (h).

- Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.

- Up to two values of L that are NOT evenly divisible by h.

- Whether or not zero-length IVs are supported.

- Whether or not a counter is used, and if so:

-- One or more of the values 8, 16, 24, 32 that equal the length of the binary representation of the counter (r).

-- Location of the counter relative to fixed input data: before, after, or in the middle.

o Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.

o Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.

o Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.

- The length (I_length) of the input values I.

For each supported combination of I_length, MAC, salt, PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I and pseudorandom salt values. If the KDF supports zero-length IVs, five of these test vectors will be accompanied by pseudorandom IVs and the other five will use zero-length IVs. If zero-length IVs are not supported, each test vector will be accompanied by an pseudorandom IV. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it.

For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

Double Pipeline Iteration Mode Tests:

The evaluator shall determine the following characteristics of the key derivation function:

- One or more pseudorandom functions that are supported by the implementation (PRF).

- The length (in bits) of the output of the PRF (h).

- Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.

- Up to two values of L that are NOT evenly divisible by h.

- Whether or not a counter is used, and if so:

-- One or more of the values 8, 16, 24, 32 that equal the length of the binary representation of the counter (r).

-- Location of the counter relative to fixed input data: before, after, or in the middle.

o Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.

o Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.

o Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.

- The length (I_length) of the input values I.

For each supported combination of I_length, MAC, salt, PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I, and pseudorandom salt values. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it.

For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

See Section 1.2 for a listing of applicable CAVP certificates.

2.2.11 Cryptographic Key Generation (MDFPP33:FCS_CKM_EXT.3)

2.2.11.1 MDFPP33:FCS_CKM_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.2 MDFPP33:FCS_CKM_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the key hierarchy section of the TSS to ensure that the formation of all KEKs are described and that the key sizes match that described by the ST author. The evaluator shall examine the key hierarchy section of the TSS to ensure that each key (DEKs, software-based key storage, and KEKs) is encrypted by keys of equal or greater security strength using one of the selected methods.

The evaluator shall review the TSS to verify that it contains a description of the conditioning used to derive KEKs. This description must include the size and storage location of salts. This activity may be performed in combination with that for FCS_COP.1/CONDITION.

(conditional) If the symmetric KEK is generated by an RBG, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1 or documentation available for the operational environment to determine that the key size being requested is greater than or equal to the key size and mode to be used for the encryption/decryption of the data.

(conditional) If the KEK is generated according to an asymmetric key scheme, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_CKM.1 is invoked. The evaluator uses the description of the key generation functionality in FCS_CKM.1 or documentation available for the operational environment to determine that the key strength being requested is greater than or equal to 112 bits.

(conditional) If the KEK is formed from a combination, the evaluator shall verify that the TSS describes the method of combination and that this method is either an XOR, a KDF, or encryption.

(conditional) If a KDF is used, the evaluator shall ensure that the TSS includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108.

(conditional) If 'concatenating the keys and using a KDF (as described in (SP 800-56C)' is selected, the evaluator shall ensure the TSS includes a description of the randomness extraction step. The description must include

- How an approved untruncated MAC function is being used for the randomness extraction step and the evaluator must verify the TSS describes that the output length (in bits) of the MAC function is at least as large as the targeted security strength (in bits) of the parameter set employed by the key establishment scheme (see Tables 1-3 of SP 800-56C).

- How the MAC function being used for the randomness extraction step is related to the PRF used in the key expansion and verify the TSS description includes the correct MAC function:

-- If an HMAC-hash is used in the randomness extraction step, then the same HMAC-hash (with the same hash function hash) is used as the PRF in the key expansion step.

-- If an AES-CMAC (with key length 128, 192, or 256 bits) is used in the randomness extraction step, then AES-CMAC with a 128-bit key is used as the PRF in the key expansion step.

- The lengths of the salt values being used in the randomness extraction step and the evaluator shall verify the TSS description includes correct salt lengths:

-- If an HMAC-hash is being used as the MAC, the salt length can be any value up to the maximum bit length permitted for input to the hash function hash.

-- If an AES-CMAC is being used as the MAC, the salt length shall be the same length as the AES key (i.e. 128, 192, or 256 bits).

The evaluator shall also ensure that the documentation of the product's encryption key management is detailed enough that, after reading, the product's key management hierarchy is clear and that it meets the requirements to ensure the keys are adequately protected. The evaluator shall ensure that the documentation includes both an essay and one or more diagrams. Note that this may also be documented as separate proprietary evidence rather than being included in the TSS.

Section 6.2 of the TSS explains the TOE generates KEKs (which are always AES 256-bit keys generated by one of the TOE's DRBGs) through a combination of methods. First, the TOE generates a KEK (the Keystore masterkey) for each user of the TOE. The TOE also generates encryption KEKs for FBE, the SD Card encryption, and work profile encryption (normal and sensitive).

The TOE generates a number of different KEKs. In addition to the TSF KEKs, applications may request key generation (through either the Android APIs or SCrypto APIs within the TEE), and the TOE utilizes its BoringSSL/SCrypto CTR_DRBG and Kernel Crypto HMAC_DRBG to satisfy those requests. The requesting application ultimately chooses whether to use that key as a DEK or a KEK, but it is worth mentioning here, as an application can utilize such a key as a KEK, should it choose.

The KMD provides the key hierarchy diagrams and a more detailed discussion of how keys are derived.

Section 1.2 contains a mapping to CAVP certificates and includes one for KDF for FBE.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: If a KDF is used, the evaluator shall perform one or more of the following tests to verify the correctness of the key derivation function, depending on the mode(s) that are supported. Table 5 maps the data fields to the notations used in SP 800-108 and SP 800-56C.

Table 5: Notations used in SP 800-108 and SP 800-56C

Data Fields	Notations	
	SP 800-108	SP 800-56C
Pseudorandom function	PRF	PRF

Counter length	r	r
Length of output of PRF	h	h
Length of derived keying material	L	L
Length of input values	I length	I length
Pseudorandom input values I	K1 (key derivation key)	Z (shared secret)
Pseudorandom salt values	n/a	s
Randomness extraction MAC	n/a	MAC

Counter Mode Tests:

The evaluator shall determine the following characteristics of the key derivation function:

- One or more pseudorandom functions that are supported by the implementation (PRF).

- One or more of the values 8, 16, 24, 32 that equal the length of the binary representation of the counter (r).

- The length (in bits) of the output of the PRF (h).

- Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.

- Up to two values of L that are NOT evenly divisible by h.

- Location of the counter relative to fixed input data: before, after, or in the middle.

-- Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.

-- Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.

-- Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.

- The length (I_length) of the input values I.

For each supported combination of I_length, MAC, salt, PRF, counter location, value of r, and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I, and pseudorandom salt values. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it. For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

Feedback Mode Tests:

The evaluator shall determine the following characteristics of the key derivation function:

- One or more pseudorandom functions that are supported by the implementation (PRF).

- The length (in bits) of the output of the PRF (h).

- Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.

- Up to two values of L that are NOT evenly divisible by h.

- Whether or not zero-length IVs are supported.

- Whether or not a counter is used, and if so:

-- One or more of the values 8, 16, 24, 32 that equal the length of the binary representation of the counter (r).

-- Location of the counter relative to fixed input data: before, after, or in the middle.

o Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.

o Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.

o Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.

- The length (I_length) of the input values I.

For each supported combination of I_length, MAC, salt, PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I and pseudorandom salt values. If the KDF supports zero-length IVs, five of these test vectors will be accompanied by pseudorandom IVs and the other five will use zero-length IVs. If zero-length IVs are not supported, each test vector will be accompanied by an pseudorandom IV. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it.

For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

Double Pipeline Iteration Mode Tests:

The evaluator shall determine the following characteristics of the key derivation function:

- One or more pseudorandom functions that are supported by the implementation (PRF).

- The length (in bits) of the output of the PRF (h).

- Minimum and maximum values for the length (in bits) of the derived keying material (L). These values can be equal if only one value of L is supported. These must be evenly divisible by h.

- Up to two values of L that are NOT evenly divisible by h.

- Whether or not a counter is used, and if so:

-- One or more of the values 8, 16, 24, 32 that equal the length of the binary representation of the counter (r).

-- Location of the counter relative to fixed input data: before, after, or in the middle.

o Counter before fixed input data: fixed input data string length (in bytes), fixed input data string value.

o Counter after fixed input data: fixed input data string length (in bytes), fixed input data string value.

o Counter in the middle of fixed input data: length of data before counter (in bytes), length of data after counter (in bytes), value of string input before counter, value of string input after counter.

- The length (I_length) of the input values I.

For each supported combination of I_length, MAC, salt, PRF, counter location (if a counter is used), value of r (if a counter is used), and value of L, the evaluator shall generate 10 test vectors that include pseudorandom input values I, and pseudorandom salt values. If there is only one value of L that is evenly divisible by h, the evaluator shall generate 20 test vectors for it.

For each test vector, the evaluator shall supply this data to the TOE in order to produce the keying material output. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

See Section 1.2 for a listing of applicable CAVP certificates.

2.2.12 Key Destruction (MDFPP33:FCS_CKM_EXT.4)

2.2.12.1 MDFPP33:FCS_CKM_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.2 MDFPP33:FCS_CKM_EXT.4.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check to ensure the TSS lists each type of plaintext key material (DEKs, software-based key storage, KEKs, trusted channel keys, passwords, etc.) and its generation and storage location.

The evaluator shall verify that the TSS describes when each type of key material is cleared (for example, on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, when transitioning to the locked state, and possibly including immediately after use, while in the locked state, etc.).

The evaluator shall also verify that, for each type of key, the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite with random pattern, or block erase) is listed. If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored.

Samsung provided a proprietary table in Section 2.1 of the KMD that lists each type of plaintext key material by its name (type and size in bits), its origin, its storage location, when it is cleared and what type of clearing. Section 6.2 of the ST states that the TOE destroys cryptographic keys when they are no longer in use by the system. The exceptions to this are public keys (that protect the boot chain and software updates) and the REK, which are never cleared. Keys stored in RAM during use are destroyed by a zero overwrite. Keys stored in Flash (i.e. eMMC) are destroyed by cryptographic erasure through a block erase call to the flash controller for the location where the FBE and SD Card keys are stored. Once these are erased, all keys (and data) stored within the encrypted data partition of the TOE are considered cryptographically erased.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

For each software and firmware key clearing situation (including on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, when transitioning to the locked state, and possibly including immediately after use, while in the locked state) the evaluator shall repeat the following tests.

For these tests the evaluator shall utilize appropriate development environment (e.g. a Virtual Machine) and development tools (debuggers, simulators, etc.) to test that keys are cleared, including all copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.

Test 3: Applied to each key held as plaintext in volatile memory and subject to destruction by overwrite by the TOE (whether or not the plaintext value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator shall:

1. Record the value of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Cause the TOE to stop the execution but not exit.

5. Cause the TOE to dump the entire memory of the TOE into a binary file.

6. Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.

7. Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece.

Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

Step 7 ensures that partial key fragments do not remain in memory. If a fragment is found, there is a minuscule chance that it is not within the context of a key (e.g., some random bits that happen to match). If this is the case the test should be repeated with a different key in Step #1. If a fragment is found the test fails.

Test 4: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:

1. Record the value of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Search the non-volatile memory the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.

5. Break the key value from Step #1 into 3 similar sized pieces and perform a search using each piece. If a fragment is found then the test is repeated (as described for test 1 above), and if a fragment is found in the repeated test then the test fails.

Test 5: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:

1. Record the storage location of the key in the TOE subject to clearing.

2. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.

3. Cause the TOE to clear the key.

4. Read the storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.

The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.

Test 3 – The evaluator received an engineering build from Samsung. The evaluator then used that build to run a series of memory dump tests that dumped the memory on the device. The evaluator took the memory dumps and searched those dumps with a hex search tool to search for known keys. The evaluator was unable to find any of the keys in the dump files.

Test 4 – Not applicable. This test requires destruction of plaintext keys and describes the applicable method. None of the keys stored in flash is stored in plaintext so it appears they are not subject to this test. Furthermore, this test applies to cases where keys are subject to destruction by overwrite and the flash-based encrypted keys in this case are subject to cryptographic erasure.

Test 5 – See test case 2 for rationale why this is not applicable.

2.2.13 TSF Wipe (MDFPP33:FCS_CKM_EXT.5)

2.2.13.1 MDFPP33:FCS_CKM_EXT.5.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.13.2 MDFPP33:FCS_CKM_EXT.5.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check to ensure the TSS describes how the device is wiped, the type of clearing procedure that is performed (cryptographic erase or overwrite) and, if overwrite is performed, the overwrite procedure (overwrite with zeros, overwrite three or more times by a different alternating pattern, overwrite with random pattern, or block erase).

If different types of memory are used to store the data to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, data stored on flash are cleared by overwriting once with zeros, while data stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write).

Section 6.2 of the ST states the TOE provides a TOE Wipe function that first erases the encrypted DEKs used to encrypt the data partition using a block erase and read verify command to ensure that the UFS blocks containing the encrypted DEKs (FBE and SD card) are now reported as empty. After the encrypted keys have been erased, the TOE will delete the entire user partition with a block erase command and then reformat the partition. Upon completion of reformatting the Flash partition holding user data, the TOE will perform a power-cycle.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance describes how to enable encryption, if it is not enabled by default. Additionally the evaluator shall verify that the AGD guidance describes how to initiate the wipe command.

Section 2.2 of the Admin Guide states the mobile device automatically encrypts data on the internal flash media of the device using AES 256. Section 8.2 discusses the wipe procedure.

Component Testing Assurance Activities: Evaluation Activity Note: The following test may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

Test 6: The evaluator shall perform one of the following tests. The test before and after the wipe command shall be identical. This test shall be repeated for each type of memory used to store the data to be protected.

Test 6.1 for File-based Methods:

The evaluator shall enable encryption according to the AGD guidance. The evaluator shall create a user data (protected data or sensitive data) file, for example, by using an application. The evaluator shall use a tool provided by the developer to examine this data stored in memory (for example, by examining a decrypted files). The evaluator shall initiate the wipe command according to the AGD guidance provided for FMT_SMF.1. The evaluator shall use a tool provided by the developer to examine the same data location in memory to verify that the data has been wiped according to the method described in the TSS (for example, the files are still encrypted and cannot be accessed).

Test 6.2 for Volume-based Methods:

The evaluator shall enable encryption according to the AGD guidance. The evaluator shall create a unique data string, for example, by using an application. The evaluator shall use a tool provided by the developer to search decrypted data for the unique string. The evaluator shall initiate the wipe command according to the AGD guidance provided for FMT_SMF.1. The evaluator shall use a tool provided by the developer to search for the same unique string in decrypted memory to verify that the data has been wiped according to the method described in the TSS (for example, the files are still encrypted and cannot be accessed).

Test 7: The evaluator shall cause the device to wipe and verify that the wipe concludes with a power cycle.

Test 6.1 (for File-based Methods) – The evaluator encrypted the internal device storage and SD card. The evaluator then wrote a known pattern to the internal storage and SD card. Next the evaluator wiped the device and dumped the contents of the internal storage and SD card. The evaluator found the contents of internal storage and SD card to be wiped.

Test 6.2 (for Volume-based methods) – Not applicable. The TOE uses file-based encryption and not volume-based encryption.

Test 7 – As part of test 6.1, the evaluator verified that the wipe concludes with a power cycle.

2.2.14 Salt Generation (MDFPP33:FCS_CKM_EXT.6)

2.2.14.1 MDFPP33:FCS_CKM_EXT.6.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS contains a description regarding the salt generation, including which algorithms on the TOE require salts. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.1. For PBKDF derivation of KEKs, this evaluation activity may be performed in conjunction with FCS_CKM_EXT.3.2.

The ST, section 6.2, explains the TOE creates salt and nonces (which are just salt values used in WPA3/WPA2) using its AES-256 CTR_DRBG. For each place that required a salt value (algorithms, PBKDF derivation of KEKs) the evaluator verified the salt was included in the description.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.15 Bluetooth Key Generation (BT10:FCS_CKM_EXT.8)

2.2.15.1 BT10:FCS_CKM_EXT.8.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS describes the criteria used to determine the frequency of generating new ECDH public/private key pairs. In particular, the evaluator shall ensure that the implementation does not permit the use of static ECDH key pairs.

Section 6.2 of the ST states the TOE will generate new ECDH key pairs for every pairing attempt.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following steps:

Step 1: Pair the TOE to a remote Bluetooth device and record the public key currently in use by the TOE. (This public key can be obtained using a Bluetooth protocol analyzer to inspect packets exchanged during pairing.)

Step 2: Perform necessary actions to generate new ECDH public/private key pairs. (Note that this test step depends on how the TSS describes the criteria used to determine the frequency of generating new ECDH public/private key pairs.)

Step 3: Pair the TOE to a remote Bluetooth device and again record the public key currently in use by the TOE.

Step 4: Verify that the public key in Step 1 differs from the public key in Step 3.

Test - The evaluator set up each of the TOE devices one at a time to snoop Bluetooth connections and then advertise for Bluetooth. For each TOE device, the evaluator used a test device to repeatedly attempt to pair with the TOE device – the attempts were alternately accepted and rejected (cancelled on the TOE device). The test device was unpaired immediately after every successful pairing. After several attempts were concluded, the Bluetooth log was collected from the TOE device. The evaluator found that the public keys were different in every case indicating that the public key pairs change for every pairing attempt.

2.2.16 Cryptographic Operation (MDFPP33:FCS_COP.1/CONDITION)

2.2.16.1 MDFPP33:FCS_COP.1.1/CONDITION

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check that the TSS describes the method by which the password is first encoded and then fed to the SHA algorithm and verify the SHA algorithm matches the first selection.

If a key stretching function, such as PBKDF2, is selected the settings for the algorithm (padding, blocking, etc.) shall be described. The evaluator shall verify that the TSS contains a description of how the output of the hash function or key stretching function is used to form the submask that will be input into the function and is the same length as the KEK as specified in FCS_CKM_EXT.3.

If any manipulation of the key is performed in forming the submask that will be used to form the KEK, that process shall be described in the TSS.

Section 6.2 explains that The TOE conditions the user's password using a combination of functions to increase the memory required for derivation to thwart attacks. This combination of functions is embedded in the scrypt algorithm. Scrypt uses three steps to condition the password:

1. One PBKDF2 operation (NIST SP 800-132)
2. Several rounds of ROMix operations
3. One final PBKDF2 operation (NIST SP 800-132)

This value is used as input for decrypting other keys that are tied to successful user authentication.

To unlock the user's keystore, the value generated in the previous step us conditioned further using PBKDF2 (NIST SP 800-132). The key derivation function uses the value derived from the initial password conditioning and a randomly generated 128-bit salt in 8192 HMAC-SHA-256 iterations to generate a 256-bit KEK.

In both cases of conditioning, the time needed to derive keying material does not impact or lessen the difficulty faced by an attacker's exhaustive guessing as the combination of the password derived KEK with REK value entirely prevents offline attacks and the TOE's maximum incorrect password login attempts (between 1 and 30 incorrect attempts with 4 character, minimum, passwords) prevents exhaustive online attacks.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: There are no test evaluation activities for this component. No explicit testing of the formation of the submask from the input password is required.

2.2.17 Cryptographic Operation (MDFPP33:FCS_COP.1/ENCRYPT)

2.2.17.1 MDFPP33:FCS_COP.1.1/ENCRYPT

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

AES-CBC Tests

Test 8: AES-CBC Known Answer Tests

There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

Test 8.1: KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

Test 8.2: KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

Test 8.3: KAT-3. To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N].

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

Test 8.4: KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

Test 9: AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i <= 10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <= 10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

Test 10: AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

# Input: PT, IV, Key for i = 1 to 1000: if i == 1: CT[1] =

AES-CBC-Encrypt(Key, IV, PT) PT = IV else: CT[i] = AES-CBC-Encrypt(Key, PT) PT = CT[i-1]

The ciphertext computed in the 1000 iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-CCM Tests

Test 11: The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:

128 bit and 256 bit keys

Two payload lengths. One payload length shall be the shortest supported payload length, greater than or equal to zero bytes. The other payload length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits).

Two or three associated data lengths. One associated data length shall be 0, if supported. One associated data length shall be the shortest supported payload length, greater than or equal to zero bytes. One associated data length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of 2 bytes, an associated data length of 2^16 bytes shall be tested.

Nonce lengths. All supported nonce lengths between 7 and 13 bytes, inclusive, shall be tested.

Tag lengths. All supported tag lengths of 4, 6, 8, 10, 12, 14 and 16 bytes shall be tested.

To test the generation-encryption functionality of AES-CCM, the evaluator shall perform the following four tests:

Test 11.1: For EACH supported key and associated data length and ANY supported payload, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

Test 11.2: For EACH supported key and payload length and ANY supported associated data, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

Test 11.3: For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator shall supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.

Test 11.4: For EACH supported key and tag length and ANY supported associated data, payload and nonce length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

To determine correctness in each of the above tests, the evaluator shall compare the ciphertext with the result of generation-encryption of the same inputs with a known good implementation.

To test the decryption-verification functionality of AES-CCM, for EACH combination of supported associated data length, payload length, nonce length and tag length, the evaluator shall supply a key value and 15 nonce, associated data and ciphertext 3-tuples and obtain either a FAIL result or a PASS result with the decrypted payload. The evaluator shall supply 10 tuples that should FAIL and 5 that should PASS per set of 15.

AES-GCM Test

The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

Test 12: The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

Test 13: The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

XTS-AES Test

Test 14: The evaluator shall test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:

256 bit (for AES-128) and 512 bit (for AES-256) keys

Three data unit (i.e., plaintext) lengths. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.

using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

Test 15: The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.

AES Key Wrap (AES-KW) and Key Wrap with Padding (AES-KWP) Test

Test 16: The evaluator shall test the authenticated encryption functionality of AES-KW for EACH combination of the following input parameter lengths:

128 and 256 bit key encryption keys (KEKs)

Three plaintext lengths. One of the plaintext lengths shall be two semi-blocks (128 bits). One of the plaintext lengths shall be three semi-blocks (192 bits). The third data unit length shall be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).

using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator shall use the AES-KW authenticated-encryption function of a known good implementation.

Test 17: The evaluator shall test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption.

Test 18: The evaluator shall test the authenticated-encryption functionality of AES-KWP using the same test as for AES-KW authenticated-encryption with the following change in the three plaintext lengths:

- One plaintext length shall be one octet. One plaintext length shall be 20 octets (160 bits).

- One plaintext length shall be the longest supported plaintext length less than or equal to 512 octets (4096 bits).

Test 19: The evaluator shall test the authenticated-decryption functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption.

See Section 1.2 for a listing of applicable CAVP certificates.

Per NIAP Policy Letter #5 CAVP Mapping, AES-CCMP is addressed by Wi-Fi Alliance certification in addition to its AES-CCM certificate already listed in Section 1.2. See WLANC10:FCS_CKM.1 for Wi-Fi Alliance certificate numbers.

2.2.18 Cryptographic Operation (VPNC24:FCS_COP.1/ENCRYPT)

2.2.18.1 VPNC24:FCS_COP.1.1/ENCRYPT

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Refer to the EAs for FCS_COP.1/ENCRYPT in the MDF PP (included below). The only change to this SFR is that some selections are mandated, therefore the corresponding testing is mandatory. The actual testing for those selections is not changed.

The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

AES-CBC Tests

Test 1: AES-CBC Known Answer Tests

There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

Test 1.1: KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

Test 1.2: KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

Test 1.3: KAT-3. To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N].

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

Test 1.4: KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

Test 2: AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i <= 10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <= 10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

Test 3: AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
if i == 1:
CT[1] = AES-CBC-Encrypt(Key, IV, PT)
PT = IV
else:
CT[i] = AES-CBC-Encrypt(Key, PT)
```

PT = CT[i-1]

The ciphertext computed in the 1000 iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt ith AES-CBC-Decrypt.

AES-CCM Tests

Test 1: The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:

128 bit and 256 bit keys

Two payload lengths. One payload length shall be the shortest supported payload length, greater than or equal to zero bytes. The other payload length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits).

Two or three associated data lengths. One associated data length shall be 0, if supported. One associated data length shall be the shortest supported payload length, greater than or equal to zero bytes. One associated data length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of 2 bytes, an associated data length of 2^16 bytes shall be tested.

Nonce lengths. All supported nonce lengths between 7 and 13 bytes, inclusive, shall be tested.

Tag lengths. All supported tag lengths of 4, 6, 8, 10, 12, 14 and 16 bytes shall be tested.

To test the generation-encryption functionality of AES-CCM, the evaluator shall perform the following four tests:

Test 1.1: For EACH supported key and associated data length and ANY supported payload, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

Test 1.2: For EACH supported key and payload length and ANY supported associated data, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

Test 1.3: For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator shall supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.

Test 1.4: For EACH supported key and tag length and ANY supported associated data, payload and nonce length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

To determine correctness in each of the above tests, the evaluator shall compare the ciphertext with the result of generation-encryption of the same inputs with a known good implementation.

To test the decryption-verification functionality of AES-CCM, for EACH combination of supported associated data length, payload length, nonce length and tag length, the evaluator shall supply a key value and 15 nonce, associated data and ciphertext 3-tuples and obtain either a FAIL result or a PASS result with the decrypted payload. The evaluator shall supply 10 tuples that should FAIL and 5 that should PASS per set of 15.

AES-GCM Test

The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

Test 1: The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

Test 2: The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

XTS-AES Test

Test 1: The evaluator shall test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:

256 bit (for AES-128) and 512 bit (for AES-256) keys

Three data unit (i.e., plaintext) lengths. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.

using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

Test 2: The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.

AES Key Wrap (AES-KW) and Key Wrap with Padding (AES-KWP) Test

Test 1: The evaluator shall test the authenticated encryption functionality of AES-KW for EACH combination of the following input parameter lengths:

128 and 256 bit key encryption keys (KEKs)

Three plaintext lengths. One of the plaintext lengths shall be two semi-blocks (128 bits). One of the plaintext lengths shall be three semi-blocks (192 bits). The third data unit length shall be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).

using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator shall use the AES-KW authenticated-encryption function of a known good implementation.

Test 2: The evaluator shall test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption.

Test 3: The evaluator shall test the authenticated-encryption functionality of AES-KWP using the same test as for AES-KW authenticated-encryption with the following change in the three plaintext lengths:

One plaintext length shall be one octet. One plaintext length shall be 20 octets (160 bits).

One plaintext length shall be the longest supported plaintext length less than or equal to 512 octets (4096 bits).

Test 4: The evaluator shall test the authenticated-decryption functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption.

See Section 1.2 for a listing of applicable CAVP certificates.

2.2.19 Cryptographic Operation (MDFPP33:FCS_COP.1/HASH)

2.2.19.1 MDFPP33:FCS_COP.1.1/HASH

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS. The evaluator shall check that the TSS indicates if the hashing function is implemented in bit-oriented and/or byte-oriented mode.

Section 6.2 of the ST associates the hash function with HMAC and digital signature operations.

Component Guidance Assurance Activities: The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required hash sizes is present.

The Admin Guide, section 3.1, explains that when the TOE is in CC mode, it will only use approved cryptographic functions. No additional configuration is needed beyond putting the device in CC mode.

Component Testing Assurance Activities: Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP. As there are different tests for each mode, an indication is given in the following sections for the bitoriented vs. the byteoriented tests.

Test 20: Short Messages Test: Bit-oriented Mode

The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages ranges sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Test 21: Short Messages Test: Byte-oriented Mode

The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Test 22: Selected Long Messages Test: Bit-oriented Mode

The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where 1 <= i <= m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Test 23: Selected Long Messages Test: Byte-oriented Mode

The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 <= i <= m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Test 24: Pseudorandomly Generated Messages Test: Byte-oriented Mode

This test is for byte oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of SHAVS. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

See Section 1.2 for a listing of applicable CAVP certificates.

2.2.20 Cryptographic Operation (MDFPP33:FCS_COP.1/KEYHMAC)

2.2.20.1 MDFPP33:FCS_COP.1.1/KEYHMAC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

The ST, section 6.2, explains that for the HMAC implementation, the TOE accepts all key sizes of 160, 256, 384, & 512; supports all SHA sizes save 224 (e.g., SHA-1, 256, 384, & 512), utilizes the specified block size (512 for SHA-1 and 256, and 1024 for SHA-384 & 512) and output MAC lengths of 160, 256, 384, and 512.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and IV using a known good implementation.

See Section 1.2 for a listing of applicable CAVP certificates.

Per NIAP Policy Letter #5 CAVP Mapping, 802.11-2012 key generation is addressed by Wi-Fi Alliance certification and HMAC certification. The 802.11ac-2013 protocol is implemented in the wpa_supplicant. The wpa_supplicant uses BoringSSL and its CAVP number is listed in section 1.2 which includes HMAC SHA-384. See WLANC10:FCS_CKM.1 for the Wi-Fi Alliance certificate numbers.

2.2.21 Cryptographic Operation (MDFPP33:FCS_COP.1/SIGN)

2.2.21.1 MDFPP33:FCS_COP.1.1/SIGN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Test 25: [conditional] If ECDSA schemes is selected in FCS_COP.1.1/SIGN

Test 25.1: ECDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

Test 25.2: ECDSA FIPS 186-4 Signature Verification Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Test 26: [conditional] If RSA schemes is selected in FCS_COP.1.1/SIGN

Test 26.1: Signature Generation Test The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages.

The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

Test 26.2: Signature Verification Test

The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

See Section 1.2 for a listing of applicable CAVP certificates.

2.2.22 HTTPS Protocol (MDFPP33:FCS_HTTPS_EXT.1)

2.2.22.1 MDFPP33:FCS_HTTPS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.22.2 MDFPP33:FCS_HTTPS_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.22.3 MDFPP33:FCS_HTTPS_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Test 27: The evaluator shall attempt to establish an HTTPS connection with a webserver, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.

Other tests are performed in conjunction with testing in the Package for Transport Layer Functional Package for TransportLayer Security (TLS), version 1.1.

Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1, and the evaluator shall perform the following test:

Test 28: The evaluator shall demonstrate that using a certificate without a valid certification path results in an application notification. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the application is notified of the validation failure.

Test 27 – This was tested in FCS_TLSC_EXT.1.1, test case 1.

Test 28 - This was tested in FCS_TLSC_EXT.1.3, test case 1.

2.2.23 IPsec - per TD0662 (VPNC24:FCS_IPSEC_EXT.1)

Component TSS Assurance Activities: In addition to the TSS EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall perform the following:

If the TOE boundary includes a general-purpose operating system or mobile device, the evaluator shall examine the TSS to ensure that it describes whether the VPN client capability is architecturally integrated with the platform itself or whether it is a separate executable that is bundled with the platform.

Section 1.4 of the ST states the TOE is a mobile device based on Android 14 with a built-in IPsec VPN client and modifications made to increase the level of security provided to end users and enterprises.

Component Guidance Assurance Activities: In addition to the Operational Guidance EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall perform the following:

If the configuration of the IPsec behavior is from an environmental source, most notably a VPN gateway (e.g through receipt of required connection parameters from a VPN gateway), the evaluator shall ensure that the operational guidance contains any appropriate information for ensuring that this configuration can be properly applied.

Note in this case that the implementation of the IPsec protocol must be enforced entirely within the TOE boundary; i.e. it is not permissible for a software application TOE to be a graphical front-end for IPsec functionality implemented totally or in part by the underlying OS platform. The behavior referenced here is for the possibility that the configuration of the IPsec connection is initiated from outside the TOE, which is permissible so long as the TSF is solely responsible for enforcing the configured behavior. However, it is allowable for the TSF to rely on low-level platform-provided networking functions to implement the SPD from the client (e.g., enforcement of packet routing decisions).

Section 3.4.4 of the Admin Guide explains the security functions implemented in the Gateway. It explains what needs to be configured in a Gateway including encryption settings, IKE protocols, and cryptoperiod.

Component Testing Assurance Activities: As a prerequisite for performing the Test EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall do the following:

The evaluator shall minimally create a test environment equivalent to the test environment illustrated below. It is expected that the traffic generator is used to construct network packets and will provide the evaluator with the ability manipulate fields in the ICMP, IPv4, IPv6, UDP, and TCP packet headers. The evaluator shall provide justification for any differences in the test environment.

Note that the evaluator shall perform all tests using the VPN client and a representative sample of platforms listed in the ST (for TOEs that claim to support multiple platforms).

The evaluator set up a test environment as described in Section 3.4. In the test set up, the “endpoint” is configured to also have the ability to capture all traffic on the connected LAN. The TOE devices connect to an

Access Point connected to that LAN so that it is possible to capture all traffic coming and going to each TOE test device. In regard to traffic generators, the evaluator used other devices on the test LAN as necessary to generate any traffic required by each individual test case. In many cases, processes on the “endpoint” generate the necessary traffic, but there are exceptions where multiple traffic sources are necessary.

2.2.23.1 VPNC24:FCS_IPSEC_EXT.1.1

TSS Assurance Activities: The evaluator shall examine the TSS and determine that it describes how the IPsec capabilities are implemented.

If the TOE is a standalone software application, the evaluator shall ensure that the TSS asserts that all IPsec functionality is implemented by the TSF. The evaluator shall also ensure that the TSS identifies what platform functionality the TSF relies upon to support its IPsec implementation, if any (e.g. does it invoke cryptographic primitive functions from the platform's cryptographic library, enforcement of packet routing decisions by low-level network drivers).

If the TOE is part of a general-purpose desktop or mobile OS, the evaluator shall ensure that the TSS describes at a high level the architectural relationship between the VPN client portion of the TOE and the rest of the TOE (e.g. is the VPN client an integrated part of the OS or is it a standalone executable that is bundled into the OS package). If the SPD is implemented by the underlying platform in this case, then the TSS describes how the client interacts with the platform to establish and populate the SPD, including the identification of the platform's interfaces that are used by the client.

In all cases, the evaluator shall also ensure that the TSS describes how the client interacts with the network stack of the platforms on which it can run (e.g., does the client insert itself within the stack via kernel mods, does the client simply invoke APIs to gain access to network services).

The evaluator shall ensure that the TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how the available rules and actions form the SPD using terms defined in RFC 4301 such as BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in RFC 4301. As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.

Section 6.2 of the ST explains that the TOE presents as few configuration options as possible to the User in order to minimize the possibility of misconfiguration and relies upon the Gateway to enforce organizational policies (for things like the specific cipher suites). For this reason, the VPN Client does not support editing of its SPD entries.

The VPN Client will insert a PROTECT rule to IPsec encrypt and send all TOE traffic to the VPN GW (as the VPN Client ignores the IKEv2 Traffic Selector negotiated between the client and gateway and always sends all traffic). The VPN Client routes all packets through the kernel's IPsec interface (ipsec0) when the VPN is active. The kernel compares packets routed through this interface to the SPDs configured for the VPN to determine whether to PROTECT, BYPASS, or DISCARD each packet. The vendor designed the TOE's VPN Client, when operating in CC Mode, to allow no SPD configuration and always force all traffic through the VPN. The VPN Client ignores any IKEv2 traffic selector negotiations with the VPN GW and will always create an SPD PROTECT rule that matches all traffic. Thus, the kernel will match all packets, subsequently encrypt those packets, and finally forward them to the VPN Gateway.

Guidance Assurance Activities: The evaluator shall examine the operational guidance to verify it describes how the SPD is created and configured. If there is an administrative interface to the client, then the guidance describes how the administrator specifies rules for processing a packet. The description includes all three cases - a rule that ensures packets are encrypted/decrypted, dropped, and allowing a packet to flow in plaintext. The evaluator shall determine that the description in the operational guidance is consistent with the description in the TSS, and that the level of detail in the operational guidance is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

If the client is configured by an external application, such as the VPN gateway, then the operational guidance should indicate this and provide a description of how the client is configured by the external application. The description should contain information as to how the SPD is established and set up in an unambiguous fashion. The description should also include what is configurable via the external application, how ordering of entries may be expressed, as well as the impacts that ordering of entries may have on the packet processing.

In either case, the evaluator ensures the description provided In the TSS is consistent with the capabilities and description provided in the operational guidance.

Section 3.4.4 of the Admin Guide explains the security functions implemented in the Gateway. It explains what needs to be configured in a Gateway including encryption settings, IKE protocols, and cryptoperiod. As explained in the TSS AA, the TOE does not support editing of its SPD entries.

Testing Assurance Activities: Depending on the implementation, the evaluator may be required to use a VPN gateway or some form of application to configure the client. For Test 2, the evaluator is required to choose an application that allows for the configuration of the full set of capabilities of the VPN client (in conjunction with the platform). For example, if the client provides a robust interface that allows for specification of wildcards, subnets, etc., it is unacceptable for the evaluator to choose a VPN Gateway that only allows for specifying a single fully qualified IP addresses in the rule.

The evaluator shall perform the following tests:

Test 1: The evaluator shall configure an SPD on the client that is capable of the following: dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the client with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule. The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behavior: appropriate packets were dropped, allowed through without modification, was encrypted by the IPsec implementation.

Test 2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. These scenarios must exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and operational guidance. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the operational guidance.

Test 1 &2 (run together) - Since control over SPDs is limited to connecting and disconnecting the VPN client, the evaluator showed that the platform was implementing BYPASS (allowing all traffic) until the point that the VPN client was connected to a gateway and then the evaluator showed that traffic was encrypted (PROTECT). The evaluator subsequently set up pings to show that when the BYPASS rule (client disconnected) was in effect the pings were successful, but once the VPN client was connected and the pings did not fall within scope of the VPN they were subject to DISCARD (no ping responses from the TOE, showing they were discarded by the TOE).

2.2.23.2 VPNC24:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: The evaluator shall check the TSS to ensure it states that the VPN can be established to operate in tunnel mode, transport mode, or either mode (as selected).

The second paragraph of the IPsec discussion in Section 6.2 says the TOE supports IPsec in tunnel mode (as selected in the SFR).

Guidance Assurance Activities: The evaluator shall confirm that the operational guidance contains instructions on how to configure the connection in each mode selected.

If both transport mode and tunnel mode are implemented, the evaluator shall review the operational guidance to determine how the use of a given mode is specified.

Instructions on configuring a connection in tunnel mode are included in the guidance. Section 3.4 of the Admin Guide specifies the settings for setting up a VPN connection (certificate, gateway, pre-shared key).

Testing Assurance Activities: The evaluator shall perform the following test(s) based on the selections chosen:

Test 1 [conditional]: If tunnel mode is selected, the evaluator uses the operational guidance to configure the TOE to operate in tunnel mode and also configures a VPN gateway to operate in tunnel mode. The evaluator configures the TOE and the VPN gateway to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the client to connect to the VPN GW peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

Test 2 [conditional]: If transport mode is selected, the evaluator uses the operational guidance to configure the TOE to operate in transport mode and also configures an IPsec peer to accept IPsec connections using transport mode. The evaluator configures the TOE and the endpoint device to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE to connect to the remote endpoint. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.

Test 3 [conditional]: If both tunnel mode and transport mode are selected, the evaluator shall perform both Test 1 and Test 2 above, demonstrating that the TOE can be configured to support both modes.

Test 4 [conditional]: If both tunnel mode and transport mode are selected, the evaluator shall modify the testing for FCS_IPSEC_EXT.1 to include the supported mode for SPD PROTECT entries to show that they only apply to traffic that is transmitted or received using the indicated mode.

Test 1 – The evaluator configured a VPN gateway to require tunnel mode. The device successfully connected to the VPN gateway and traffic log supports tunnel mode was used for the device.

Test 2, 3, & 4 – Not applicable as the TOE supports only tunnel mode.

2.2.23.3 VPNC24:FCS_IPSEC_EXT.1.3

TSS Assurance Activities: The evaluator shall examine the TSS to verify that the TSS provides a description of how a packet is processed against the SPD and that if no 'rules' are found to match, that a final rule exists, either implicitly or explicitly, that causes the network packet to be discarded.

Section 6.2 of the ST explains that the TOE presents as few configuration options as possible to the user in order to minimize the possibility of misconfiguration and relies upon the Gateway to enforce organizational policies (for things like the specific cipher suites and selection of traffic to protect). For this reason, the TOE does not support editing of its SPD entries. The TOE will insert a PROTECT rule to IPsec encrypt and send all TOE traffic to the VPN GW (as the TOE ignores the IKEv2 Traffic Selector negotiated between the client and gateway and always sends all traffic). The TOE routes all packets through the kernel’s IPsec interface (ipsec0) when the VPN is active. The kernel compares packets routed through this interface to the SPDs configured for the VPN to determine whether to PROTECT, BYPASS, or DISCARD each packet. The vendor designed the TOE’s VPN, when operating in CC Mode, to allow no configuration and to always force all traffic through the VPN. The TOE ignores any IKEv2 traffic selector negotiations with the VPN GW and will always create an SPD PROTECT rule that matches all traffic. Thus, the kernel will match all packets, subsequently encrypt those packets, and finally forward them to the VPN Gateway.

Guidance Assurance Activities: The evaluator shall check that the operational guidance provides instructions on how to construct or acquire the SPD and uses the guidance to configure the TOE for the following test.

The TOE does not support editing of SPD entries. See the TSS assurance activity for an explanation.

Testing Assurance Activities: The evaluator shall perform the following test:

Test 1: The evaluator shall configure the SPD such that it has entries that contain operations that DISCARD, PROTECT, and (if applicable) BYPASS network packets. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches a BYPASS entry and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a 'TOE created' final entry that discards packets that do not match any previous entries). The evaluator sends the packet, and observes that the packet was not permitted to flow to any of the TOE's interfaces.

Test 1 – This was tested as part of FCS_IPSEC_EXT.1.1 test 1 where the range of SPD scenarios was covered since the only possible SPDs are bypass when the VPN client is disconnected and protect and discard when it is connected. There are no options to configure SPD rules of differing specificity or to configure rule ordering.

2.2.23.4 VPNC24:FCS_IPSEC_EXT.1.4

TSS Assurance Activities: The evaluator shall examine the TSS to verify that the algorithms AES-GCM-128 and AES-GCM-256 are implemented. If the ST author has selected either AES-CBC-128 or AES-CBC-256 in the requirement, then the evaluator verifies the TSS describes these as well. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in the relevant iteration of FCS_COP.1 from the Base-PP that applies to keyed-hash message authentication.

Section 6.2 of the ST states the TOE implements RFC 4106 conformant AES-GCM-128, and AES-GCM-256, and RFC 3602 conformant AES-CBC-128, and AES-CBC-256 as encryption algorithms. Section 6.2 states the TOE Platform also provides SHA-1, SHA-256, SHA-384, and SHA-512 in addition to HMAC-SHA1, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 as integrity/authentication algorithms (producing message digests of 160, 256, 384, and 512-bits in length).

Guidance Assurance Activities: The evaluator checks the operational guidance to ensure it provides instructions on how the TOE is configured to use the algorithms selected in this component and whether this is performed through direct configuration, defined during initial installation, or defined by acquiring configuration settings from an environmental component.

Section 3.3 of the Admin Guide provides instructions for configuring the device into CC mode. For the VPN configuration, CC Mode only needs to be Enforced as that will set the required FIPS encryption. The TOE does not need to be configured to support specific encryption algorithms. The TOE will respond with the appropriate algorithm to the VPN Gateway.

Testing Assurance Activities: Test 1: Test 1: The evaluator shall configure the TOE as indicated in the operational guidance configuring the TOE to using each of the AES-GCM-128, and AES-GCM-256 algorithms, and attempt to establish a connection using ESP. If the ST Author has selected either AES-CBC-128 or AES-CBC-256, the TOE is configured to use those algorithms and the evaluator attempts to establish a connection using ESP for those algorithms selected.

Test 1 - The evaluator made an IPsec connection to a VPN gateway using each of the claimed IPsec ESP ciphersuites (AES-GCM-128, AES-GCM-256, AES-CBC-128 and AES-CBC-256).

2.2.23.5 VPNC24:FCS_IPSEC_EXT.1.5

TSS Assurance Activities: The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented. If IKEv1 is implemented, the evaluator shall verify that the TSS indicates whether or not XAUTH is supported, and that aggressive mode is not used for IKEv1 Phase 1 exchanges (i.e. only main mode is used). It may be that these are configurable options.

Section 6.2 of the ST states that the TOE provides IKEv2 key establishment as part of its IPsec implementation.

Guidance Assurance Activities: The evaluator shall check the operational guidance to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and uses the guidance to configure the TOE to perform NAT traversal for the test below. If XAUTH is implemented, the evaluator shall verify that the operational guidance provides instructions on how it is enabled or disabled.

If the TOE supports IKEv1, the evaluator shall verify that the operational guidance either asserts that only main mode is used for Phase 1 exchanges, or provides instructions for disabling aggressive mode.

Section 3.3 of the Admin Guide provides instructions for configuring the device into CC mode. No configuration is necessary for IKEv2 and NAT traversal. The TOE supports each automatically.

Testing Assurance Activities: Test 1:

a. The evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 7296, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

b. If the TOE supports IKEv1 with or without XAUTH, the evaluator shall verify that this test can be successfully repeated with XAUTH enabled and disabled in the manner specified by the operational guidance. If the TOE only supports IKEv1 with XAUTH, the evaluator shall verify that connections not using XAUTH are unsuccessful. If the TOE only supports IKEv1 without XAUTH, the evaluator shall verify that connections using XAUTH are unsuccessful.

In the case that the VPN gateway enforces the TOE's configuration, the following steps shall be performed to meet the objective of Test 1:

1. Configure the TOE client and VPN gateway to have XAUTH enabled.

2. Attempt the connection and observe that the connection succeeds and that XAUTH is used.

3. Configure the TOE and gateway to have XAUTH disabled.

4. Attempt the connection and observe that the connection succeeds and that XAUTH is not present.

5. Attempt to configure a mismatch between the TOE and gateway (i.e. modify a local configuration setting on the client system)

6. Verify that no IPsec connection is attempted until the gateway corrects the configuration settings

Test 2: [conditional]: If the TOE supports IKEv1, the evaluator shall perform any applicable operational guidance steps to disable the use of aggressive mode and then attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator shall show that the TOE will reject a VPN gateway from initiating an IKEv1 Phase 1 connection in aggressive mode. The evaluator should then show that main mode exchanges are supported.

In the case that the VPN gateway enforces the TOE's configuration, the following steps should be performed to meet the objective of Test 2:

1. Configure the gateway and TOE client in the appropriate manner per the guidance documentation. (Gateway rejects Aggressive mode, Client rejects aggressive mode)

2. Connect the TOE client to the gateway to obtain the configuration settings.

3. Observe the main mode connection is successful.

4. Disconnect the TOE from the gateway.

5. Attempt to modify the setting for main mode locally on the TOE to force the client to attempt to use aggressive mode.

6. Observe that when the initial connection attempt to the gateway is made, the gateway detects the configuration difference and reapplies the main mode setting before the TOE can attempt an IPsec connection.

7. Configure a peer to have equivalent settings to the VPN gateway (Same ciphers/Authentication/Hash/KEX settings)

8. Tell the TOE that there is a VPN gateway at the location of the peer.

9. Observe that the TOE cannot establish a connection with the peer.

(TD0662 applied)

Test 1 – The evaluator put the device behind a NAT router. The evaluator then connected the device to the VPN gateway. The packet capture of the traffic demonstrates that the connection with the NAT router was working.

Test 2 – Not applicable. IKEv1 is not supported.

2.2.23.6 VPNC24:FCS_IPSEC_EXT.1.6

TSS Assurance Activities: The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms AES-CBC-128, AES-CBC-256 are specified, and if others are chosen in the selection of the requirement, those are included in the TSS discussion.

Section 6.2 of the ST states the encrypted payload for IKEv2 uses AES-CBC-128 and AES-CBC-256 as specified in RFC 6379 and AES-GCM-128 and AES-GCM-256 as specified in RFC 5282.

Guidance Assurance Activities: The evaluator checks the operational guidance to ensure it provides instructions on how the TOE is configured to use the algorithms selected in this component and whether this is performed through direct configuration, defined during initial installation, or defined by acquiring configuration settings from an environmental component.

Section 3.3 of the Admin Guide provides instructions for configuring the device into CC mode. The TOE does not need to be configured to support specific encryption algorithms. The TOE will respond with the appropriate algorithm to the VPN Gateway.

Testing Assurance Activities: The evaluator shall use the operational guidance to configure the TOE (or to configure the Operational Environment to have the TOE receive configuration) to perform the following test for each ciphersuite selected:

Test 1: The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation. The evaluator will confirm that the connection is successful by confirming that data can be passed through the connection once it is established. For example, the evaluator may connect to a webpage on the remote network and verify that it can be reached.

Test 1 - The evaluator made an IPsec connection to a VPN gateway using each of the claimed IKE ciphersuites. The evaluator was able to capture each ciphersuite using a packet capture.

2.2.23.7 VPNC24:FCS_IPSEC_EXT.1.7

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall check the operational guidance to ensure it provides instructions on how the TOE configures the values for SA lifetimes. In addition, the evaluator shall check that the guidance has the option for either the Administrator or VPN Gateway to configure Phase 1 SAs if time-based limits are supported. Currently there are no values mandated for the number of packets or number of bytes, the evaluator shall simply check the operational guidance to ensure that this can be configured if selected in the requirement.

Section 3.4.4 of the Admin Guide explains that there are many configuration options for a VPN tunnel that can only be configured from the gateway. The VPN client will utilize these settings from the gateway configuration to construct the secure tunnel. The section has an entry for IPsec Session Key cryptoperiod and recommends that the gateway specifies the session key cryptoperiod and can be used to configure periods under 1 hour in duration.

Testing Assurance Activities: When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC 'A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered.'

Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

Test 1 [conditional]: The evaluator shall configure a maximum lifetime in terms of the # of packets (or bytes) allowed following the operational guidance. The evaluator shall establish an SA and determine that once the allowed # of packets (or bytes) through this SA is exceeded, the connection is closed.

Test 2 [conditional]: The evaluator shall construct a test where a Phase 1 SA is established and attempted to be maintained for more than 24 hours before it is renegotiated. The evaluator shall observe that this SA is closed or renegotiated in 24 hours or less. If such an action requires that the TOE be configured in a specific way, the evaluator shall implement tests demonstrating that the configuration capability of the TOE works as documented in the operational guidance.

Test 3 [conditional]: The evaluator shall perform a test similar to Test 2 for Phase 2 SAs, except that the lifetime will be 8 hours or less instead of 24 hours or less.

Test 4 [conditional]: If a fixed limit for IKEv1 SAs is supported, the evaluator shall establish an SA and observe that the connection is closed after the fixed traffic and/or time value is reached.

Test 1 – This test is not applicable because that packet quantity option was not selected in the requirement.

Test 2 – The evaluator configured a Phase 1 SA lifetime timeout of 25 hours on the VPN server. The evaluator then established a connection with the VPN. The IPsec SA timed out after just under 8 hours and the connection reset.

Test 3 - The evaluator configured a Phase 2 SA lifetime timeout of 9 hours for on the VPN server. The evaluator then established a connection with the VPN. The IPsec SA timed out after just under 3 hours and the connection reset.

Test 4 – This test was performed in test 2 where both Phase 1 and Phase 2 SAs are tested together.

2.2.23.8 VPNC24:FCS_IPSEC_EXT.1.8

TSS Assurance Activities: The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

Section 6.2 of the ST states the TOE supports the following DH groups: 14, 19, 20 and 24. The TOE selects the DH group by selecting the largest group configured by an administrator that is offered by the VPN gateway.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform the following test:

Test 1: For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols can be successfully completed using that particular DH group.

Test 1 - The evaluator made an IPsec connection to a VPN gateway using each of the claimed DH groups (14, 19, 20 and 24). The evaluator was able to capture each DH group using a packet capture.

2.2.23.9 VPNC24:FCS_IPSEC_EXT.1.9

TSS Assurance Activities: The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating 'x' (as defined in FCS_IPSEC_EXT.1.9) and each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this EP is used, and that the length of 'x' and the nonces meet the stipulations in the requirement.

Section 6.2 states the nonce is generated using a FIPS validated RBG with lengths of 224, 256, and 384 bits. The TSS also has all supported DH groups listed along with their group values (e.g., 14 (2048-bit MODP), 19 (256-bit Random ECP)).

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.23.10 VPNC24:FCS_IPSEC_EXT.1.10

TSS Assurance Activities: Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.9.

See VPNC24:FCS_IPSEC_EXT.1.9

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.23.11 VPNC24:FCS_IPSEC_EXT.1.11

TSS Assurance Activities: The evaluator shall ensures that the TSS whether peer authentication is performed using RSA, ECDSA, or both.

If any selection with pre-shared keys is chosen in the selection, the evaluator shall check to ensure that the TSS describes how those selections work in conjunction with authentication of IPsec connections.

The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include whether the certificate presented identifier is compared to the ID payload presented identifier, which fields of the certificate are used as the presented identifier (DN, Common Name, or SAN) and, if multiple fields are supported, the logical order comparison. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer's presented certificate.

Section 6.2 of the ST states that the TOE implements peer authentication using RSA certificates or ECDSA certificates that conform to RFC 4945. The FCS_COP.1/SIGN SFR does support both RSA and ECDSA certificates.

Section 6.2 of the ST states, the VPN Client ensures that that the Subject Alternative Name IP address or DNS name in the VPN gateway’s certificate matches the IP address or DNS name configured in the VPN profile and ensures that the certificate has not been revoked (using the Online Certificate Status Protocol [OCSP] in accordance with RFC 2560).

Guidance Assurance Activities: If any selection with 'Pre-shared Keys' is selected, the evaluator shall check that the operational guidance describes any configuration necessary to enable any selected authentication mechanisms.

If any method other than no other method is selected, the evaluator shall check that the operational guidance describes any configuration necessary to enable any selected authentication mechanisms.

The evaluator ensures the operational guidance describes how to set up the TOE to use the cryptographic algorithms RSA, ECDSA, or either, depending which is claimed in the ST.

In order to construct the environment and configure the TOE for the following tests, the evaluator will ensure that the operational guidance also describes how to configure the TOE to connect to a trusted CA, and ensure a valid certificate for that CA is loaded into the TOE as a trusted CA.

The evaluator shall also ensure that the operational guidance includes the configuration of the reference identifiers for the peer.

Section 3.3 of the Admin Guide provides a worksheet that contains a tab for VPN which contains the settings for selecting the cryptographic algorithm in the VPN Type setting. Section 3.4.1 further explains that RSA and ECDSA are supported with IKEv2. Section 3.3 identifies the settings for selecting certificates. It explains how to import certificates into the Trust Anchor Database and how to import and remove certificates.

Testing Assurance Activities: For efficiency's sake, the testing that is performed here has been combined with the testing for FIA_X509_EXT.2 and FIA_X509_EXT.3 (for IPsec connections and depending on the Base-PP), FCS_IPSEC_EXT.1.12, and FCS_IPSEC_EXT.1.13. The following tests shall be repeated for each peer authentication protocol selected in the FCS_IPSEC_EXT.1.11 selection above:

Test 1: The evaluator shall have the TOE generate a public-private key pair, and submit a CSR (Certificate Signing Request) to a CA (trusted by both the TOE and the peer VPN used to establish a connection) for its signature. The values for the DN (Common Name, Organization, Organizational Unit, and Country) will also be passed in the request. Alternatively, the evaluator may import to the TOE a previously generated private key and corresponding certificate.

Test 2: The evaluator shall configure the TOE to use a private key and associated certificate signed by a trusted CA and shall establish an IPsec connection with the peer.

Test 3: The evaluator shall test that the TOE can properly handle revoked certificates - conditional on whether CRL or OCSP is selected; if both are selected, and then a test is performed for each method. For this current version of the PP-Module, the evaluator has to only test one up in the trust chain (future drafts may require to ensure the validation is done up the entire chain). The evaluator shall ensure that a valid certificate is used, and that the SA is established. The evaluator then attempts the test with a certificate that will be revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the TOE will not establish an SA.

Test 4 [conditional]: For each selection made, the evaluator shall verify factors are required, as indicated in the operational guidance, to establish an IPsec connection with the server.

For each supported identifier type (excluding DNs), the evaluator shall repeat the following tests:

Test 5: For each field of the certificate supported for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds.

Test 6: For each field of the certificate support for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to not match the field in the peer's presented certificate and shall verify that the IKE authentication fails.

The following tests are conditional:

Test 7 [conditional]: If, according to the TSS, the TOE supports both Common Name and SAN certificate fields and uses the preferred logic outlined in the Application Note, the tests above with the Common Name field shall be performed using peer certificates with no SAN extension. Additionally, the evaluator shall configure the peer's reference identifier on the TOE to not match the SAN in the peer's presented certificate but to match the Common Name in the peer's presented certificate, and verify that the IKE authentication fails.

Test 8 [conditional]: If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKE authentication succeeds. To demonstrate a bit-wise comparison of the DN, the evaluator shall change a single bit in the DN (preferably, in an Object Identifier (OID) in the DN) and verify that the IKE authentication fails. To demonstrate a comparison of DN values, the evaluator shall change any one of the four DN values and verify that the IKE authentication fails.

Test 9 [conditional]: If the TOE supports both IPv4 and IPv6 and supports IP address identifier types, the evaluator must repeat test 1 and 2 with both IPv4 address identifiers and IPv6 identifiers. Additionally, the evaluator shall verify that the TOE verifies that the IP header matches the identifiers by setting the presented identifiers and the reference identifier with the same IP address that differs from the actual IP address of the peer in the IP headers and verifying that the IKE authentication fails.

Test 10 [conditional]: If, according to the TSS, the TOE performs comparisons between the peer's ID payload and the peer's certificate, the evaluator shall repeat the following test for each combination of supported identifier types and supported certificate fields (as above). The evaluator shall configure the peer to present a different ID payload than the field in the peer's presented certificate and verify that the TOE fails to authenticate the IKE peer.

Test 1 – This test was performed in FIA_X509_EXT.1 test case 1 for IPsec where a certificate had been configured prior to a successful connection in each case.

Test 2 - This test was performed in FIA_X509_EXT.1 test case 1 for IPsec where a certificate (and associated private key) had been configured and used to successfully connect after its certificate chain was validated.

Test 3 - This test was performed in FIA_X509_EXT.1 test case 3 for IPsec where certificate revocation is tested for the peer certificate as well as its issuing chain certificates.

Test 4 – Not applicable – pre-shared keys are not claimed.

Test 5 - This test was performed in FCS_IPSEC_EXT.1.11 test case 6 where IP and FQDN are used for connections where the identifiers match and do not match.

Test 6 - The evaluator iteratively configured a test peer to use an authentication certificate with the correct and incorrect IP address and correct and incorrect DNS address. For the first iteration, the IKE ID matched the certificate IP Address and the connection was established as expected. For the second iteration, the IKE ID did not match the certificate IP Address and the connection was rejected as expected. For the third iteration, the IKE ID matched the certificate FQDN Address and the connection was established as expected. For the fourth iteration, the IKE ID did not match the certificate FQDN Address and the connection was rejected as expected. This test was performed using ECDSA-IKEv2 and RSA-IKEv2.

Test 7 - Not applicable since only SAN matching is supported.

Test 8 - Not applicable since the TOE does not support DN identifier types.

Test 9 - Not applicable since the TOE supports only IPv4.

Test 10 - Not applicable since the TOE does not support matching the peer identity with the peer certificate.

2.2.23.12 VPNC24:FCS_IPSEC_EXT.1.12

TSS Assurance Activities: Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.11.

See VPNC24:FCS_IPSEC_EXT.1.11

Guidance Assurance Activities: Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.11.

See VPNC24:FCS_IPSEC_EXT.1.11.

Testing Assurance Activities: Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.11.

See VPNC24:FCS_IPSEC_EXT.1.11.

2.2.23.13 VPNC24:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.11.

See VPNC24:FCS_IPSEC_EXT.1.11.

Guidance Assurance Activities: Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.11.

See VPNC24:FCS_IPSEC_EXT.1.11.

Testing Assurance Activities: Assurance Activities for this element are tested through Assurance Activities for FCS_IPSEC_EXT.1.11.

See VPNC24:FCS_IPSEC_EXT.1.11.

2.2.23.14 VPNC24:FCS_IPSEC_EXT.1.14

TSS Assurance Activities: The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.

Section 6.2 of the ST states the TOE implements RFC 4106 conformant AES-GCM-128 and AES-GCM-256, and RFC 3602 conformant AES-CBC-128 and AES-CBC-256 as encryption algorithms. The TOE implements HMAC-SHA1, SHA-256, SHA-384, and SHA-512 as authentication algorithms as well as Diffie-Hellman Groups 14, 19, 20 and 24. The encrypted payload for IKEv2 uses AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and AES-GCM-128 and AES-GCM-256 as specified in RFC 5282. The TOE relies upon the VPN Gateway to ensure that the cryptographic algorithms and key sizes negotiated during the IKEv2 negotiation ensure that the security strength of the Phase 1/IKE_SA are greater than or equal to that of the Phase 2/CHILD_SA.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator follows the guidance to configure the TOE to perform the following tests:

Test 1: This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.

Test 2 [conditional]: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.

Test 3: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.

Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.

Test 1 - All of the algorithms were already tested in FCS_IPSEC_EXT.1.6. The evaluator focused on the hashes for this test and configured the VPN gateway to request each hash (one at a time). The evaluator then connected the device using each claimed hash successfully.

Test 2 - This property is not enforced by the TOE but rather must be enforced by the VPN gateway since it determines the applicable cipher strengths. As such there is no applicable test.

Test 3 - The evaluator configured the VPN gateway to use an unallowed cipher. The evaluator then attempted to connect the device with the VPN gateway. The connection was refused because the unallowed cipher is not supported.

Test 4 – The evaluator configured the VPN gateway to use the unallowed cipher to establish an SA for ESP. The evaluator then attempted to connect the device with the VPN gateway. The connection was refused because the unallowed cipher is not supported to establish an SA for ESP.

2.2.24 Initialization Vector Generation (MDFPP33:FCS_IV_EXT.1)

2.2.24.1 MDFPP33:FCS_IV_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the key hierarchy section of the TSS to ensure that the encryption of all keys is described and the formation of the IVs for each key encrypted by the same KEK meets FCS_IV_EXT.1.

The ST, section 6.2, describes the process of generating IVs for data storage and key storage encryption. The TOE uses AES-XTS and AES-CBC mode for data encryption and AES-GCM for key storage.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.25 Random Bit Generation - per TD0677 (MDFPP33:FCS_RBG_EXT.1)

2.2.25.1 MDFPP33:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.25.2 MDFPP33:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.25.3 MDFPP33:FCS_RBG_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: Entropy Documentation

Documentation shall be produced and the evaluator shall perform the activities in accordance with Appendix F - Entropy Documentation And Assessment, the 'Clarification to the Entropy Documentation and Assessment'.

API Documentation

The evaluator shall verify that the API documentation provided according to Section 5.2.2 Class ADV: Development, includes the security functions described in FCS_RBG_EXT.1.3.

The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality.

Section 6.2 of the ST explains that the TOE provides a number of different RBGs including:

1. An AES-256 CTR_DRBG provided by BoringSSL. The TOE provides mobile applications access (through an Android API) to random data drawn from its AES-256 CTR_DRBG
2. An AES-256 CTR_DRBG provided by SCrypto in the TEE
3. A SHA-256 HMAC_DRBG provided by Kernel Crypto in the Android kernel (/dev/random or get_random_bytes())
4. A hardware SHA-256 Hash_DRBG provided by the Qualcomm Application Processor hardware

The TOE ensures that it initializes each RBG with sufficient entropy ultimately accumulated from a TOE-hardware-based noise source. The TOE uses its hardware-based noise source to fill primary input pool continuously with random data that has full entropy, and in turn, the TOE draws from this input pool to seed both its AES-256 CTR_DRBG and its SHA-256 HMAC_DRBG. The TOE seeds each of its software DRBGs using 384-bits of data from the primary input pool, thus ensuring at least 256-bits of entropy. These RBGs are all capable of providing other amounts of entropy (such as 128-bits) to any requesting application. The TOE itself always uses 256-bits of entropy, but other applications or services are not subject to this limitation, and can request 128-bits or 256-bits of entropy subject its own requirements. The SHA-256 Hash_DRBG in the Qualcomm AP is used to generate the REK on first boot. Note that the entropy analysis has been accepted by CCEVS/NIAP.

Section 6.1 of the Admin Guide contains the Random Number API. This section contains an API for applications to get random data.

Component Testing Assurance Activities: Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. 'generate one block of random bits' means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

Entropy input: the length of the entropy input value must equal the seed length.

Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.

Personalization string: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

See Section 1.2 for a listing of applicable CAVP certificates.

2.2.26 Random Bit Generator State Preservation (MDFPP33:FCS_RBG_EXT.2)

2.2.26.1 MDFPP33:FCS_RBG_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluation activity for this requirement is captured in the RBG documentation for Appendix F - Entropy Documentation And Assessment. The evaluator shall verify that the documentation describes how the state is generated so as to be available for the next startup, how the state is used as input to the DRBG, and any protection measures used for the state while the TOE is powered off.

Section 6.2 of the ST explains the devices save a block of 4096-bits of random data, and upon the next boot, a service called entropy mixer adds the block of saved random data into the Linux Kernel Random Number Generator’s input pool.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.27 Cryptographic Algorithm Services (MDFPP33:FCS_SRV_EXT.1)

2.2.27.1 MDFPP33:FCS_SRV_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall verify that the API documentation provided according to Section 5.2.2 Class ADV: Development includes the security functions (cryptographic algorithms) described in these requirements.

Section 6.1 of the Admin Guide contains the Cryptographic APIs. There is a specific API for each cryptographic algorithm in the ST.

Component Testing Assurance Activities: The evaluator shall write, or the developer shall provide access to, an application that requests cryptographic operations by the TSF. The evaluator shall verify that the results from the operation match the expected results according to the API documentation. This application may be used to assist in verifying the cryptographic operation Evaluation Activities for the other algorithm services requirements.

Test – The developer provided a test program that invoked each of the cryptographic APIs in the documentation. The evaluator mapped the service SFRs with the APIs used in the test program and verified the test program against the documentation to ensure completeness and correctness

2.2.28 Cryptographic Algorithm Services (MDFPP33:FCS_SRV_EXT.2)

2.2.28.1 MDFPP33:FCS_SRV_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall verify that the API documentation for the secure key storage includes the cryptographic operations by the stored keys.

Section 6.1 of the Admin Guide contains the Cryptographic APIs. There is a specific API for each cryptographic algorithm in the ST.

Component Testing Assurance Activities: The evaluator shall verify that the API documentation for the secure key storage includes the cryptographic operations by the stored keys.

The evaluator shall write, or the developer shall provide access to, an application that requests cryptographic operations of stored keys by the TSF. The evaluator shall verify that the results from the operation match the expected results according to the API documentation. The evaluator shall use these APIs to test the functionality of the secure key storage according to the Evaluation Activities in FCS_STG_EXT.1.

Test – This was tested as part of MDFPP33:FCS_SRV_EXT.1.1 where all of the cryptographic operations are tested including encryption/decryption and digital signing per the identified requirements.

2.2.29 Cryptographic Key Storage (MDFPP33:FCS_STG_EXT.1)

2.2.29.1 MDFPP33:FCS_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.29.2 MDFPP33:FCS_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.29.3 MDFPP33:FCS_STG_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.29.4 MDFPP33:FCS_STG_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.29.5 MDFPP33:FCS_STG_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall review the TSS to determine that the TOE implements the required secure key storage. The evaluator shall ensure that the TSS contains a description of the key storage mechanism that justifies the selection of 'mutable hardware' or 'software-based'.

Section 6.2 of the ST explains the TOE provides users, administrators and applications running on the TOE the ability to generate, import, and securely store symmetric and asymmetric keys through the TOE's Android Keystore. The TOE allows a user or administrator (via the MDM) to import a certificate (in PKCS#12 [PFX] format) and provides applications running on the TOE an API to import a certificate or secret key. In either case, the TOE will place the key into the user's keystore (and the TOE will remove the PKCS#12 password-based protection if the imported key is a certificate) and doubly encrypt the imported key with DEKs, which in turn are encrypted by a KEK derived from the user's DKEK and a KEK derived from the REK. All user and application keys placed into the user's keystore are secured in this fashion.

The user of the TOE can elect to delete keys from the keystore, as well as to wipe the entire device securely. The administrator can only delete keys from the keystore that have been deployed to the TOE by the administrator.

The TOE affords applications control (control over use and destruction) of keys that they create or import, and only the common application developer can explicitly authorize access, use, or destruction of one application's key by any other application.

On the devices with support for mutable hardware storage (see Table 17 - Mutable Key Storage Components), the TOE provides both software-based (the Android Keystore) and mutable hardware-based (the StrongBox Keystore) key storage. The key storage is implemented in a separate hardware chipset that is part of the SoC, providing additional protection beyond that normally provided by the Android Keystore which is implemented in the TEE. The hardware module implements its own cryptographic services internally to the module.

Device	Mutable Storage Chipset
All S24FE(Samsung)	Smart Secure Platform 7.0
All ZFlip6 (Qualcomm)	Secure Processing Unit 1.3

Device	Mutable Storage Chipset
All S24FE(Samsung)	Smart Secure Platform 7.0
All ZFlip6 (Qualcomm)	Secure Processing Unit 1.3

Table 17 - Mutable Key Storage Components

By default, all key storage is handled with the Android Keystore, but an application developer can use the StrongBox keystore by setting a preference to use the hardware keystore instead. When the preference is set to the hardware keystore, the keys will be stored in the mutable hardware and all operations on the keys will be handled inside the hardware

Component Guidance Assurance Activities: The evaluator shall verify that the API documentation provided according to Section 5.2.2 Class ADV: Development includes the security functions (import, use, and destruction) described in these requirements. The API documentation shall include the method by which applications restrict access to their keys or secrets in order to meet FCS_STG_EXT.1.4.

The evaluator shall review the AGD guidance to determine that it describes the steps needed to import or destroy keys or secrets.

The table entry “Certificate Management” in Section 3.3 (Device Common Criteria Settings) of the Admin Guide explains how to import and remove certificates. Section 8.2 of the Admin Guide explains how to wipe a device.

Section 6.1 of the Admin Guide contains the Cryptographic APIs. These APIs address importing, using and destroying private keys. Section 6.1 explains developers can utilize the KeyStore or the KeyChain to store their keys/credentials, depending on type of key (symmetric keys can only be stored in the KeyStore). Keys stored in the KeyStore can only be accessed (used or deleted) by the original app or by apps with a common developer with enforcement handled by the KeyStore. Keys stored in the KeyChain can be made globally available (with explicit approval by the user). When a key is imported/created it is assigned authorizations for use which cannot be changed later (i.e. what the key can be used for, how long the key can be available).

Component Testing Assurance Activities: The evaluator shall test the functionality of each security function:

Test 29: The evaluator shall import keys or secrets of each supported type according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that generates a key or secret of each supported type and calls the import functions. The evaluator shall verify that no errors occur during import.

Test 30: The evaluator shall write, or the developer shall provide access to, an application that uses an imported key or secret:

- For RSA, the secret shall be used to sign data.

- For ECDSA, the secret shall be used to sign data

In the future additional types will be required to be tested:

- For symmetric algorithms, the secret shall be used to encrypt data.

- For persistent secrets, the secret shall be compared to the imported secret.

The evaluator shall repeat this test with the application-imported keys or secrets and a different application's imported keys or secrets. The evaluator shall verify that the TOE requires approval before allowing the application to use the key or secret imported by the user or by a different application:

- The evaluator shall deny the approvals to verify that the application is not able to use the key or secret as described.

- The evaluator shall repeat the test, allowing the approvals to verify that the application is able to use the key/secret as described.

If the ST author has selected 'common application developer', this test is performed by either using applications from different developers or appropriately (according to API documentation) not authorizing sharing.

Test 31: The evaluator shall destroy keys or secrets of each supported type according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that destroys an imported key or secret.

The evaluator shall repeat this test with the application-imported keys or secrets and a different application's imported keys or secrets. The evaluator shall verify that the TOE requires approval before allowing the application to destroy the key or secret imported by the administrator or by a different application:

- The evaluator shall deny the approvals and verify that the application is still able to use the key or secret as described.

- The evaluator shall repeat the test, allowing the approvals and verifying that the application is no longer able to use the key or secret as described.

If the ST author has selected 'common application developer', this test is performed by either using applications from different developers or appropriately (according to API documentation) not authorizing sharing.

Test 29 – The developer provided a program that tests generated and imported certificates. This test operates on both RSA and ECDSA keys. The developer provided a test script that generates several test applications. The applications use RSA keys and ECDSA keys. The applications generate and import cases for both RSA and ECDSA keys and attempt to access keys from apps with the same and different developer signatures. The test demonstrates only applications with the same developer can access/destroy a key.

Test 30 – See test case 29 which also addresses the access to imported and generated keys across applications.

Test 31 - See test case 29 which also addresses the destruction of imported and generated keys across applications.

2.2.30 Encrypted Cryptographic Key Storage (MDFPP33:FCS_STG_EXT.2)

2.2.30.1 MDFPP33:FCS_STG_EXT.2.1

TSS Assurance Activities: The evaluator shall review the TSS to determine that the TSS includes key hierarchy description of the protection of each DEK for data-at-rest, of software-based key storage, of long-term trusted channel keys, and of KEK related to the protection of the DEKs, long-term trusted channel keys, and software-based key storage. This description must include a diagram illustrating the key hierarchy implemented by the TOE in order to demonstrate that the implementation meets FCS_STG_EXT.2. The description shall indicate how the functionality described by FCS_RBG_EXT.1 is invoked to generate DEKs (FCS_CKM_EXT.2), the key size (FCS_CKM_EXT.2 and FCS_CKM_EXT.3) for each key, how each KEK is formed (generated, derived, or combined according to FCS_CKM_EXT.3), the integrity protection method for each encrypted key (FCS_STG_EXT.3), and the IV generation for each key encrypted by the same KEK (FCS_IV_EXT.1). More detail for each task follows the corresponding requirement.

The evaluator shall also ensure that the documentation of the product's encryption key management is detailed enough that, after reading, the product's key management hierarchy is clear and that it meets the requirements to ensure the keys are adequately protected. The evaluator shall ensure that the documentation includes both an essay and one or more diagrams. Note that this may also be documented as separate proprietary evidence rather than being included in the TSS.

Section 2.1 of the KMD includes a key hierarchy diagram that is proprietary. The TOE provides protection for all stored keys (i.e. those written to storage media for persistent storage) chained to both the user’s password and the REK. All keys are encrypted with AES-GCM or AES-CBC (in the case of SD card File Encryption Keys). All KEKs are 256-bit, ensuring that the TOE encrypts every key with another key of equal or greater strength/size.

In the case of Wi-Fi, the TOE utilizes the 802.11-2012 KCK and KEK keys to unwrap (decrypt) the WPA3/WPA2 Group Temporal Key received from the access point. Additionally, the TOE protects persistent Wi-Fi keys (user certificates) by storing them in the Android Keystore.

The TOE also stores the SecurityLogAgent (properties related to the SE Android configuration) in the same manner as the long-term trusted channel key materials.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.30.2 MDFPP33:FCS_STG_EXT.2.2

TSS Assurance Activities: The evaluator shall examine the key hierarchy description in the TSS section to verify that each DEK and software-stored key is encrypted according to FCS_STG_EXT.2.

The ST, section 6.2, describes the protection of DEKs and KEKs in detail. DEK and KEK generation is fully described earlier in section 6.2, but the MDFPP33:FCS_STG_EXT.2 subsection summarizes the earlier descriptions to address the required points.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.31 Integrity of Encrypted Key Storage (MDFPP33:FCS_STG_EXT.3)

2.2.31.1 MDFPP33:FCS_STG_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.31.2 MDFPP33:FCS_STG_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the key hierarchy description in the TSS section to verify that each encrypted key is integrity protected according to one of the options in FCS_STG_EXT.3.

The evaluator shall also ensure that the documentation of the product's encryption key management is detailed enough that, after reading, the product's key management hierarchy is clear and that it meets the requirements to ensure the keys are adequately protected. The evaluator shall ensure that the documentation includes both an essay and one or more diagrams. Note that this may also be documented as separate proprietary evidence rather than being included in the TSS.

The ST, section 6.2, explains AES-256 GCM is used to encrypt all KEKs other than SDCard keys (which uses HMAC for integrity) and the GCM encryption mode itself ensures integrity as authenticated decryption operations fail if the encrypted KEK becomes corrupted.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.32 TLS Protocol (PKGTLS11:FCS_TLS_EXT.1)

2.2.32.1 PKGTLS11:FCS_TLS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall ensure that the selections indicated in the ST are consistent with selections in the dependent components.

Section 5 of the ST contains TLS sections are consistent with selections in the dependent components.

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.33 TLS Client Protocol (PKGTLS11:FCS_TLSC_EXT.1)

2.2.33.1 PKGTLS11:FCS_TLSC_EXT.1.1

TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The evaluator shall check the TSS to ensure that the cipher suites specified include those listed for this component.

The ST, section 6.2, references the SFR for the list of ciphersuites. The ST states that no configuration is necessary for selecting the ciphersuites. The TOE inherently (without requiring any configuration) supports the supported ciphersuites and evaluated elliptic curves (P-256 and P-384); neither the user nor the administrator need configure anything in order for the TOE to support these curves.

Guidance Assurance Activities: The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the product so that TLS conforms to the description in the TSS.

No configuration is necessary to use the claimed ciphersuites.

Testing Assurance Activities: The evaluator shall also perform the following tests:

Test 1: The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 2: The goal of the following test is to verify that the TOE accepts only certificates with appropriate values in the extendedKeyUsage extension, and implicitly that the TOE correctly parses the extendedKeyUsage extension as part of X.509v3 server certificate validation. The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator shall repeat this test using a different, but otherwise valid and trusted, certificate that lacks the Server Authentication purpose in the extendedKeyUsage extension and ensure that a connection is not established. Ideally, the two certificates should be similar in structure, the types of identifiers used, and the chain of trust.

Test 3: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected cipher suite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites.) The evaluator shall verify that the product disconnects after receiving the server's Certificate handshake message.

Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.

Test 5: The evaluator shall perform the following modifications to the traffic:

Test 5.1: Change the TLS version selected by the server in the Server Hello to an undefined TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.

Test 5.2: Change the TLS version selected by the server in the Server Hello to the most recent unsupported TLS version (for example 1.1 represented by the two bytes 03 02) and verify that the client rejects the connection.

Test 5.3: [conditional] If DHE or ECDHE cipher suites are supported, modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client does not complete the handshake and no application data flows.

Test 5.4: Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client does not complete the handshake and no application data flows.

Test 5.5: [conditional] If DHE or ECDHE cipher suites are supported, modify the signature block in the server's Key Exchange handshake message, and verify that the client does not complete the handshake and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

Test 5.6: Modify a byte in the Server Finished handshake message, and verify that the client does not complete the handshake and no application data flows.

Test 5.7: Send a message consisting of random bytes from the server after the server has issued the Change Cipher Spec message and verify that the client does not complete the handshake and no application data flows. The message must still have a valid 5-byte record header in order to ensure the message will be parsed as TLS.

These tests are repeated for HTTPS and TLS.

Test 1: The evaluator established a TLS session using the interfaces stated above for each of the claimed ciphersuites in turn. The evaluator used a network sniffer to capture the TLS session negotiation and observed that the expected TLS cipher is negotiated.

Test 2: The evaluator established a TLS session using the interfaces stated above. The evaluator configured the server to send a certificate with the Server Authentication purpose in the extendedKeyUsage field. Using a network sniffer the evaluator captured the TLS session negotiation and observed that the TLS session was successfully negotiated. The evaluator reconfigured the test server to retry the TLS session using a certificate that is missing the Server Authentication purpose in the extendedKeyUsage field. Using a network sniffer the evaluator captured the TLS session negotiation and observed that the TLS session is not successfully negotiated.

Test 3: The evaluator established a TLS session using the interfaces stated above. A modified test server negotiates an ECDSA ciphersuite, but returns an RSA certificate. Using a network sniffer to capture the TLS session negotiation and observed that the TLS session is not negotiated successfully.

Test 4: The evaluator configured a test server to accept only the TLS_NULL_WITH_NUL_NULL ciphersuite. The evaluator then attempted to establish a TLS session using the interfaces stated above to that test server. Using a network sniffer the evaluator captured the TLS session negotiation and observed that the TLS session is not successfully negotiated.

Test 5: The evaluator obtained a packet captures of the TLS session negotiation using the interfaces stated above and a test server with Mutual Authentication configured on the test server. The evaluator made connection attempts from the client to the test server. The server implementation of the TLS protocol was modified as stated in the 7 scenarios described by the Assurance Activity. The evaluator inspected each packet captures to ensure that the connections are rejected for each scenario.

2.2.33.2 PKGTLS11:FCS_TLSC_EXT.1.2

TSS Assurance Activities: The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the application-configured reference identifier, including which types of reference identifiers are supported (e.g. Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the product.

The ST, section 6.2, states the TOE supports Common Name (CN) and Subject Alternative Name (SAN) (DNS and IP address) as reference identifiers. The TOE supports client (mutual) authentication. The TOE inherently (without requiring any configuration) supports the supported ciphersuites and evaluated elliptic curves (P-256 and P-384); neither the user nor the administrator need configure anything in order for the TOE to support these curves. The TOE supports the use of wildcards in X.509 reference identifiers (CN and SAN).

Guidance Assurance Activities: The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.

Section 3.3 of the Admin Guide provides a spreadsheet with includes the MDM setting for importing allowed CA certificates. Section 6.3 of the Admin Guide provides a reference to the developer information for TLS/HTTPS. Within those links is information regarding setting the reference identifier.

Testing Assurance Activities: The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection. If the TOE supports certificate pinning, all pinned certificates must be removed before performing Tests 1 through 6. A pinned certificate must be added prior to performing Test 7. (TD0499 applied)

Test 1: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails. Note that some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

Test 2: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.

Test 3: [conditional] If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension.

The evaluator shall verify that the connection succeeds. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

Test 4: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.

Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier. The support for wildcards is intended to be optional. If wildcards are supported, the first, second, and third tests below shall be executed. If wildcards are not supported, then the fourth test below shall be executed.

Test 5.1: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.

Test 5.2: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.

Test 5.3: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label immediately preceding the public suffix (e.g. *.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.com) and verify that the connection fails.

Test 5.4: [conditional]: If wildcards are not supported, the evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection fails.

Test 6: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

Test 7: [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

These tests are repeated for HTTPS and TLS.

Test 1: The evaluator attempted a connection with a valid CN and SAN and the connection was accepted. The evaluator attempted to connect to a server with a bad CN and no SAN. The connection was rejected.

Test 2: The evaluator attempted to connect to a server with a server certificate that contains a CN that matches the reference identifier and contains a bad SAN extension. The connection was rejected.

Test 3: The evaluator attempted to connect to a server with a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The connection was rejected as expected because the TOE requires SAN always.

Test 4: The evaluator attempted to connect to a server with a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The connection succeeded.

Test 5: The evaluator attempted to connect to a server with a server certificate that contains various wildcard combinations. The connections were rejected.

Test 6: The TOE does not support the optional URI or Service Name used as reference identifiers.

Test 7: The evaluator configured a server to present the certificate pinned within a test application on the TOE to ensure a successful connection could be made and then the evaluator presented a certificate that was not pinned but otherwise valid to show that a connection could not be made.

2.2.33.3 PKGTLS11:FCS_TLSC_EXT.1.3

TSS Assurance Activities: If the selection for authorizing override of invalid certificates is made, then the evaluator shall ensure that the TSS includes a description of how and when user or administrator authorization is obtained. The evaluator shall also ensure that the TSS describes any mechanism for storing such authorizations, such that future presentation of such otherwise-invalid certificates permits establishment of a trusted channel without user or administrator action.

The selection for override was not made.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that using an invalid certificate results in the function failing as follows, unless excepted:

Test 1a: The evaluator shall demonstrate that a server using a certificate with a valid certification path successfully connects.

Test 1b: The evaluator shall modify the certificate chain used by the server in test 1a to be invalid and demonstrate that a server using a certificate without a valid certification path to a trust store element of the TOE results in an authentication failure.

Test 1c [conditional]: If the TOE trust store can be managed, the evaluator shall modify the trust store element used in Test 1a to be untrusted and demonstrate that a connection attempt from the same server used in Test 1a results in an authentication failure.

(TD0513 applied)

Test 2: The evaluator shall demonstrate that a server using a certificate which has been revoked results in an authentication failure.

Test 3: The evaluator shall demonstrate that a server using a certificate which has passed its expiration date results in an authentication failure.

Test 4: The evaluator shall demonstrate that a server using a certificate which does not have a valid identifier results in an authentication failure.

These tests are repeated for HTTPS and TLS.

Test 1: The evaluator configured server using a certificate with a valid certification path terminating in a certificate which was not configured in the TOE as trusted. The evaluator observed that the TOE rejected the certificate. The evaluator then loaded the trusted CA certificate(s) needed to validate the server's certificate, and demonstrated that the connection succeeded. The evaluator then deleted the CA certificate that was loaded in the previous test part, and showed that the connection again failed.

Test 2: This test has been performed in MDFPP33:FIA_X509_EXT.1 test case 62.

Test 3: This test has been performed in MDFPP33:FIA_X509_EXT.1 test case 61.

Test 4: This test has been performed in PKGTLS11: FCS_TLSC_EXT.1.2.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.34 TLS Client Protocol (EAP-TLS for WLAN) (WLANC10:FCS_TLSC_EXT.1/WLAN)

2.2.34.1 WLANC10:FCS_TLSC_EXT.1.1/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.34.2 WLANC10:FCS_TLSC_EXT.1.2/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.34.3 WLANC10:FCS_TLSC_EXT.1.3/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.34.4 WLANC10:FCS_TLSC_EXT.1.4/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.34.5 WLANC10:FCS_TLSC_EXT.1.5/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The evaluator shall check the TSS to ensure that the cipher suites specified include those listed for this component.

The ST, section 6.2, references the SFR for the list of ciphersuites. The ST states that no configuration is necessary for selecting the ciphersuites. The TOE inherently (without requiring any configuration) supports the supported ciphersuites and evaluated elliptic curves (P-256 and P-384) and requires/allows no configuration of the supported curves.

Component Guidance Assurance Activities: The evaluator shall check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of cipher suites advertised by the TOE may have to be restricted to meet the requirements).

The evaluator shall check that the guidance contains instructions for the administrator to configure the list of Certificate Authorities that are allowed to sign certificates used by the authentication server that will be accepted by the TOE in the EAP-TLS exchange, and instructions on how to specify the algorithm suites that will be proposed and accepted by the TOE during the EAP-TLS exchange.

The Admin Guide, section 3.1, explains that Wi-Fi connections can utilize both FIPS and non-FIPS algorithms for compatibility reasons. To ensure the use of FIPS-validated algorithms in Wi-Fi connections, the Wi-Fi Access Point should be configured to specify the proper ciphersuites. Section 3.3 of the Admin Guide provides a spreadsheet which includes the MDM setting for importing allowed CA certificates.

Component Testing Assurance Activities: The evaluator shall write, or the TOE developer shall provide, an application for the purposes of testing TLS.

The evaluator shall perform the following tests:

Test 1: The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.

Test 3: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected cipher suite. For example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites. The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.

Test 5: The evaluator shall perform the following modifications to the traffic:

- Change the TLS version selected by the server in the Server Hello to a unsupported TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.

- Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE cipher suite) or that the server denies the client's Finished handshake message.

- Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.

- [conditional: if the TOE supports at least one cipher suite that uses DHE or ECDHE for key exchange] Modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message. This test does not apply to cipher suites using RSA key exchange.

- Modify a byte in the Server Finished handshake message, and verify that the client sends an Encrypted Message followed by a FIN and ACK message. This is sufficient to deduce that the TOE responded with a Fatal Alert and no further data would be sent.

- Send a garbled message from the server after the server has issued the ChangeCipherSpec message and verify that the client denies the connection.

Test 1 – The evaluator established a connection with each ciphersuite specified in the ST. The evaluator verified the correctness of the ciphersuite using the server logging options (the evaluator has a packet capture as well).

Test 2 – The evaluator configured an Access Point to use an Authentication Server for EAP-TLS authentication. The evaluator used a known good certificate that included the extendedKeyUsage field and verified the connection was accepted. The evaluator then created a server certificate without the Server Authentication purpose in the extendedKeyUsage field and verified the connection was rejected.

Test 3 – The evaluator configured an Access Point to use an Authentication Server for EAP-TLS authentication. The evaluator then attempted to use an RSA certificate while using an ECDSA ciphersuite. The connection was disconnected.

Test 4 - The evaluator configured an Access Point to use EAP-TLS authentication. The evaluator then attempted to require the TLS_NULL_WITH_NULL_NULL ciphersuite. That attempt was rejected by the client.

Test 5 – The evaluator configured an Access Point for EAP-TLS authentication. The evaluator then created packets that modified each of the required options. In all cases the connection was denied as expected.

2.2.35 TLS Client Support for Mutual Authentication (PKGTLS11:FCS_TLSC_EXT.2)

2.2.35.1 PKGTLS11:FCS_TLSC_EXT.2.1

TSS Assurance Activities: The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication. The evaluator shall also ensure that the TSS describes any factors beyond configuration that are necessary in order for the client to engage in mutual authentication using X.509v3 certificates.

Section 6.2 of the ST states the TOE supports mutual (client) authentication. The ST says no other configuration beyond a certificate is required to support the ST claims.

Guidance Assurance Activities: The evaluator shall ensure that the AGD guidance includes any instructions necessary to configure the TOE to perform mutual authentication. The evaluator also shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

Section 3.3 of the Admin Guide provides a spreadsheet with includes the MDM setting for how to import certificates.

Testing Assurance Activities: The evaluator shall also perform the following tests:

Test 1: The evaluator shall establish a connection to a server that is not configured for mutual authentication (i.e. does not send Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE did not send Client's Certificate message (type 11) during handshake.

Test 2: The evaluator shall establish a connection to a server with a shared trusted root that is configured for mutual authentication (i.e. it sends Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE responds with a non-empty Client's Certificate message (type 11) and Certificate Verify (type 15) message.

This test is repeated for TLS and HTTPS.

Test 1: The evaluator established a TLS session using a test server that was not configured for mutual authentication. The evaluator observed that the TLS connection was successful and the TOE did not send a certificate or a certificate verify message.

Test 2: The evaluator established a TLS session using a test server that was configured for mutual authentication. The evaluator observed that the TLS connection was successful and the TOE did send both a certificate and a certificate verify message.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.36 TLS Client Support for Supported Groups Extension (EAP-TLS for WLAN) (WLANC10:FCS_TLSC_EXT.2/WLAN)

2.2.36.1 WLANC10:FCS_TLSC_EXT.2.1/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the Supported Groups extension and whether the required behavior is performed by default or may be configured.

The ST states in section 6.2 that the TOE, by design, supports only evaluated elliptic curves (P-256 & P-384) and requires/allows no configuration of the supported curves.

Component Guidance Assurance Activities: If the TSS indicates that the Supported Groups extension must be configured to meet the requirement, the evaluator shall verify that the operational guidance includes instructions for configuration of this extension.

The Admin Guide, section 3.1, explains that Wi-Fi connections can utilize both FIPS and non-FIPS algorithms for compatibility reasons. To ensure the use of FIPS-validated algorithms in Wi-Fi connections, the Wi-Fi Access Point should be configured to specify the proper ciphersuites.

Component Testing Assurance Activities: The evaluator shall perform the following test:

Test 1: The evaluator shall configure a server to perform ECDHE key exchange using each of the TOE's supported curves and shall verify that the TOE successfully connects to the server.

Test 1- The evaluator configured the test server for each curve claimed by the ST. The client was able to establish a connection with each claimed curve.

2.2.37 TLS Client Support for Renegotiation (PKGTLS11:FCS_TLSC_EXT.4)

2.2.37.1 PKGTLS11:FCS_TLSC_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall use a network packet analyzer/sniffer to capture the traffic between the two TLS endpoints. The evaluator shall verify that either the 'renegotiation_info' field or the SCSV cipher suite is included in the ClientHello message during the initial handshake.

Test 2: The evaluator shall verify the Client's handling of ServerHello messages received during the initial handshake that include the 'renegotiation_info' extension. The evaluator shall modify the length portion of this field in the ServerHello message to be non-zero and verify that the client sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.

Test 3: The evaluator shall verify that ServerHello messages received during secure renegotiation contain the 'renegotiation_info' extension. The evaluator shall modify either the 'client_verify_data' or 'server_verify_data' value and verify that the client terminates the connection.

Test 1 – The evaluator configured the TOE to connect to a test server using TLS. The evaluator configured the server to connect normally with renegotiation. The evaluator observed the 'renegotiation_info' field in the handshake as expected.

Test 2 - The evaluator configured the TOE to connect to a test server using TLS. The evaluator configured the server to connect normally with renegotiation. The evaluator modified the length field of the ServerHello to be non-zero and the connection failed. The evaluator tested with an unmodified connection and the TOE connected as expected.

Test 3 - The evaluator configured the TOE to connect to a test server using TLS. The evaluator configured the server to connect normally with renegotiation. The evaluator modified server_verify_data field and the connection failed. The evaluator tested with an unmodified connection and the TOE connected as expected.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.38 TLS Client Support for Supported Groups Extension (PKGTLS11:FCS_TLSC_EXT.5)

2.2.38.1 PKGTLS11:FCS_TLSC_EXT.5.1

TSS Assurance Activities: The evaluator shall verify that TSS describes the Supported Groups Extension.

The ST states in section 6.2 that the TOE, by design, supports only evaluated elliptic curves (P-256 & P-384) and requires/allows no configuration of the supported curves.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall also perform the following test:

Test 1: The evaluator shall configure a server to perform key exchange using each of the TOE's supported curves and/or groups. The evaluator shall verify that the TOE successfully connects to the server.

Test 1- The evaluator configured the test server for each curve claimed by the ST. The client was able to establish a connection with each claimed curve.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.39 Supported WPA Versions - per TD0710 (WLANC10:FCS_WPA_EXT.1)

2.2.39.1 WLANC10:FCS_WPA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall ensure that the AGD contains guidance on how to configure the WLAN client to connect to networks supporting WPA3 and, In Section 4.2 of the Admin Guide, the user is instructed that detailed instructions for connecting to Wi-Fi networks can be found under the “Connections” section of the guide for the specific device

In Section 4.2 of the Admin Guide, the user is instructed that detailed instructions for connecting to Wi-Fi networks can be found under the “Connections” section of the guide for the specific device. Additionally, the table in Section 3.3 of the Admin Guide provides the Common Criteria Related Settings.

- The Wi-Fi Settings entry in the table includes the API for configuring the security policy for each wireless network including setting the WLAN CA certificate, specifying the security type, selecting the authentication protocol and selecting the client credentials.

Component Testing Assurance Activities: The evaluator shall configure a Wi-Fi network that utilizes WPA3 and verify that the client can connect. The same test shall be repeated for WPA2 if it is selected. (TD0710 applied)

The evaluator configured a Wi-Fi network to use WPA3 and demonstrated the TOE can connect. The evaluator repeated the test for WPA2

2.3 User data protection (FDP)

2.3.1 Access Control for System Services (MDFPP33:FDP_ACF_EXT.1)

2.3.1.1 MDFPP33:FDP_ACF_EXT.1.1

TSS Assurance Activities: The evaluator shall ensure the TSS lists all system services available for use by an application. The evaluator shall also ensure that the TSS describes how applications interface with these system services, and means by which these system services are protected by the TSF.

The TSS shall describe which of the following categories each system service falls in:

1. No applications are allowed access

2. Privileged applications are allowed access

3. Applications are allowed access by user authorization

4. All applications are allowed access

Privileged applications include any applications developed by the TSF developer. The TSS shall describe how privileges are granted to third-party applications. For both types of privileged applications, the TSS shall describe how and when the privileges are verified and how the TSF prevents unprivileged applications from accessing those services.

For any services for which the user may grant access, the evaluator shall ensure that the TSS identifies whether the user is prompted for authorization when the application is installed, or during runtime. The evaluator shall ensure that the operational user guidance contains instructions for restricting application access to system services.

Section 6.3 of the ST provides a description of the categories of system services to applications. It states that for older applications (those targeting Android's pre-23 API level, i.e., API level 22 [Android 5.0] and below), the TOE will prompt a user at the time of application installation whether they agree to grant the application access to the requested services. Thereafter (each time the application is run), the TOE will grant the application access to the services specified during install. For newer applications (those targeting API level 23 or later), the TOE grants individual permissions at application run-time by prompting the user for confirmation of each permissions category requested by the application (and only granting the permission if the user chooses to grant it). Section 6.3 also provides a description of each of the different types of permissions. The KMD provides a mapping of each system service to a privilege level.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Evaluation Activity Note: The following tests require the vendor to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

The evaluator shall write, or the developer shall provide, applications for the purposes of the following tests.

Test 32: For each system service to which no applications are allowed access, the evaluator shall attempt to access the system service with a test application and verify that the application is not able to access that system service.

Test 33: For each system service to which only privileged applications are allowed access, the evaluator shall attempt to access the system service with an unprivileged application and verify that the application is not able to access that system service. The evaluator shall attempt to access the system service with a privileged application and verify that the application can access the service.

Test 34: For each system service to which the user may grant access, the evaluator shall attempt to access the system service with a test application. The evaluator shall ensure that either the system blocks such accesses or prompts for user authorization. The prompt for user authorization may occur at runtime or at installation time, and should be consistent with the behavior described in the TSS.

Test 35: For each system service listed in the TSS that is accessible by all applications, the evaluator shall test that an application can access that system service.

Test 32 – No permissions (or services) were identified where no application would have access. Rather, there are permissions available to users, permissions available to users (but requiring user permission during app install), and permission available only to the system. These are all covered in the next three test cases.

Test 33 – The evaluator used several programs provided by the developer. Those applications are used to exercise APIs and check permissions to determine the specific permissions for each API.

Test 34 – This was tested as part of test 33 where all services are tested with and without user-granted privileges.

Test 35 - See Test Case 33 where all services are tested with and without necessary privileges.

2.3.1.2 MDFPP33:FDP_ACF_EXT.1.2

TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes which data sharing is permitted between applications, which data sharing is not permitted, and how disallowed sharing is prevented. It is possible to select both 'applications' and 'groups of applications', in which case the TSS is expected to describe the data sharing policies that would be applied in each case.

Section 6.9 of the ST explains that the TOE through a combination of Android's multi-user capabilities and Security Enhancements (SE) for Android, provides the ability to create an isolated profile within the device. Within a work profile a group of applications can be installed, and access to those applications is then restricted to usage solely within the work profile. The work profile boundary restricts the ability of sharing data such that applications outside the work profile cannot see, share or even copy data to those inside the work profile and vice versa. Exceptions to the boundary (such as allowing a copy operation) must be configured by the administrator via policy. Furthermore, the work profile boundary policy can control access to hardware features, such as the camera or microphone, and restrict the ability of applications within the work profile to access those services.

Further, Section 6.3 of the ST states that only applications with a common application developer are able to allow sharing of data between the applications. Common applications are those signed by a common certificate or key by the developer that have permissions to allow data sharing in their manifest. Application data can only be shared under this scenario.

The TOE provides the ability to create a Knox Separated Apps group. A Knox Separated Apps group provides an administrator with the ability to isolate applications from the broader system. Access to applications placed into the group do not require separate authentication. Applications within the group are restricted from accessing services provided outside the group, such as sharing services (intents) and data storage.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test 36: The evaluator shall write, or the developer shall provide, two applications, one that saves data containing a unique string and the other, which attempts to access that data. If groups of applications is selected, the applications shall be placed into different groups. If application is selected, the evaluator shall install the two applications. If private is selected, the application shall not write to a designated shared storage area. The evaluator shall verify that the second application is unable to access the stored unique string.

If the user is selected, the evaluator shall grant access as the user and verify that the second application is able to access the stored unique string.

If the administrator is selected, the evaluator shall grant access as the administrator and verify that the second application is able to access the stored unique string.

If a common application developer is selected, the evaluator shall grant access to an, application with a common application developer to the first, and verify that the application is able to access the stored unique string.

Test - The evaluator used three applications - two signed by the same developer and have the same UID and one signed using a different key so it has a different UID. The evaluator demonstrated the first two applications can share data since they share a UID. The evaluator then demonstrated the third application cannot access the shared data because it has a different UID.

Knox - The evaluator used two applications - one designed to attempt to write known data into several file locations (fixed and virtualized) and another to attempt to read those values. The evaluator exercised the tool to attempt to write data outside a Knox container finding that it could only write to expected locations (i.e., outside the container) and was denied writing files inside the Knox container. The evaluator repeated that while inside a KNOX container. Again, the evaluator found that the application could only write into expected locations (i.e., inside the Knox container) and could not write into locations outside the Knox container. The evaluator then exercised the tool to attempt to read those locations from inside the Knox container. Again, the evaluator found that it could read only locations inside the Knox container and in the case of the virtualized location that it retrieved the expected value. The evaluator repeated that outside the Knox container. As expected, the evaluator found that the application could only read locations outside the container and got the expected values, including the value from the virtualized location.

The evaluator went on to use the File Manager to attempt to move a file into the Knox container from outside. This operation failed. That was repeated inside the Knox container and again failed. The evaluator changed the container policy using available administrator functions to allow copies into and out of the Knox container. The evaluator repeated the move attempts described above and found that files could now be moved into and out of the Knox container.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.2 Access Control for System Resources (MDFPP33:FDP_ACF_EXT.2)

2.3.2.1 MDFPP33:FDP_ACF_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: For each selected resource, the evaluator shall cause data to be placed into the Enterprise group's instance of that shared resource. The evaluator shall install an application into the Personal group that attempts to access the shared resource information and verify that it cannot access the information.

Test - The evaluator used the contacts and appointments (calendar) as Knox applications and saved data. The evaluator then used those same applications in the Personal workspace and showed the Knox data was not available. The evaluator then enabled sharing between Knox and personal data for those applications and then showed that the base device could read all Knox data and that the Knox container is able to read the calendar data.

2.3.3 Security Attribute Based Access Control (MDFPP33:FDP_ACF_EXT.3)

2.3.3.1 MDFPP33:FDP_ACF_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

Test 118: The evaluator shall write, or the developer shall provide, an application that attempts to store a file with both write and execute permissions. If the selection is 'no exceptions', then the evaluator shall verify that this action fails and that the permissions on the file are not simultaneously write and execute. If the selection is 'application's private data folder', then the evaluator shall ensure that the attempt to store the file is outside of the application's private data folder.

Test 119: The evaluator shall traverse the file system examining the permission on each TSF file to verify that no file has both write and execute permissions set. If the selection is 'application's private data folder', then only files outside of this folder need to be examined by the evaluator for this test.

Test 118: The evaluator created a file in a shared directory. The evaluator then tried to change the permission to write and execute on the file. This attempted operation failed.

Test 119: The evaluator dumped the entire file system and permissions of each TOE device into a file and examined the permissions. The evaluator concludes that none of the TSF files are writeable (see MDFPP33:FPT_AEX_EXT.4, test 95)

2.3.4 Protected Data Encryption (MDFPP33:FDP_DAR_EXT.1)

2.3.4.1 MDFPP33:FDP_DAR_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.4.2 MDFPP33:FDP_DAR_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS section of the ST indicates which data is protected by the DAR implementation and what data is considered TSF data. The evaluator shall ensure that this data includes all protected data.

Section 6.3 of the ST states the TOE provides encryption of all data (which includes both user data and TSF data) stored on the data partition and on external media (such as an SD Card) of the TOE.

The TOE uses FBE to encrypt data using XTS-AES-256 using a unique File Content Encryption Key (FCEK) for each file. File metadata (such as filenames) is encrypted separately with AES-CBC-CTS with a unique File Name Encryption Key (FNEK) for each file. FBE supports two separate classes of protection, credentialed and device. While each class is encrypted, the difference is whether the encryption is chained to the user’s credentials. By default, all data is stored in the credential class, and applications that will store data in the device class are defined during the installation of the application. Device class data can be accessed as soon as the device has started, including prior to the first user authentication.

For the protection of data stored on external media (SD Card[1]), the TOE also provides AES-256-CBC encryption of protected data stored using FEKs. The TOE encrypts each individual file stored on the SD Card, generating a unique FEK for each file.

The TOE's system executables, libraries, and their configuration data reside in a read-only file system outside the data partition.

Component Guidance Assurance Activities: The evaluator shall review the AGD guidance to determine that the description of the configuration and use of the DAR protection does not require the user to perform any actions beyond configuration and providing the authentication credential. The evaluator shall also review the AGD guidance to determine that the configuration does not require the user to identify encryption on a per-file basis.

The devices are encrypted by default. Section 2.2 of the Admin Gude states the mobile device automatically encrypts data on the device using AES 256. Note that external storage is manually encrypted.

Component Testing Assurance Activities: Evaluation Activity Note: The following test requires the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

Test 37: The evaluator shall enable encryption according to the AGD guidance. The evaluator shall create user data (non-system) either by creating a file or by using an application. The evaluator shall use a tool provided by the developer to verify that this data is encrypted when the product is powered off, in conjunction with Test 1 for FIA_UAU_EXT.1.

Test 37 – The evaluator encrypted the sdcard storage through the Settings on the TOE (the internal storage is always encrypted). The evaluator then executed a script that created 3 files on the internal storage. The evaluator used the device's file viewer to ensure the files were created with the expected content on the internal storage. The evaluator then put the device in recovery mode and searched the device. The evaluator was not able to find the known data from the 3 files that were created on the internal storage. The evaluator also pulled the SD cards from the devices and found them unreadable (random seeming data) on Windows.

2.3.5 Sensitive Data Encryption (MDFPP33:FDP_DAR_EXT.2)

2.3.5.1 MDFPP33:FDP_DAR_EXT.2.1

TSS Assurance Activities: The evaluator shall verify that the TSS includes a description of which data stored by the TSF (such as by native applications) is treated as sensitive. This data may include all or some user or enterprise data

[1] Galaxy A53 and A52 devices support removable media.

and must be specific regarding the level of protection of email, contacts, calendar appointments, messages, and documents.

The evaluator shall examine the TSS to determine that it describes the mechanism that is provided for applications to use to mark data and keys as sensitive. This description shall also contain information reflecting how data and keys marked in this manner are distinguished from data and keys that are not (for instance, tagging, segregation in a 'special' area of memory or container, etc.).

Section 6.3 explains that for S24 FE and Z Flip6 5G and their equivalent devices, the TOE provides the NIAPSEC library for Sensitive Data Protection (SDP) that application developers must use to opt-in for sensitive data protection. When developer’s opt-in for SDP, all data that is received on the device destined for that application is treated as sensitive. This library calls into the TOE to generate an RSA key that acts as a master KEK for the SDP encryption process. When an application that has opted-in for SDP receives incoming data while the device is locked, an AES symmetric DEK is generated to encrypt that data. The public key from the master RSA KEK above is then used to encrypt the AES DEK. Once the device is unlocked, the RSA KEK private key is re-derived and can be used to decrypt the AES DEK for each piece of information that was stored while the device was locked. The TOE then takes that decrypted data and re-encrypts it following FDP_DAR_EXT.1.

For all other devices, the TOE as a part of the Knox Platform for Enterprise, provides mobile applications the ability to store sensitive data and have the TOE encrypt it accordingly. This functionality is controlled by the administrator through the apps that support sensitive data storage, and is only available for apps that have been designed to support sensitive data. When an application stores data as sensitive data, the file will be marked as sensitive in the metadata. Based on the environment lock-state, sensitive data protected by this mechanism will be encrypted when locked (either directly by the user or via timeout). An application can determine whether sensitive data should remain encrypted in this manner or if it should be re-encrypted (such as by a symmetric key for better performance). Applications can use this to receive and store data securely while the environment is locked (such as an email application). Details about specific keys are located in the KMD in Section 2.2.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall enable encryption of sensitive data and require user authentication according to the AGD guidance. The evaluator shall try to access and create sensitive data (as defined in the ST and either by creating a file or using an application to generate sensitive data) in order to verify that no other user interaction is required.

Test 1 - See Test Case MDFPP33:FIA_UAU_EXT.1, test 3 where data was accessed that was created from an NIAPSEC app (on the S24 FE and Z Flip6 5G devices) and within the Knox container on the rest of the devices (i.e., to create the test files). The NIAPSEC application operated to create sensitive data using the NIAPSEC library, while any app could be used in the Knox container to create sensitive data automatically without any user intervention.

2.3.5.2 MDFPP33:FDP_DAR_EXT.2.2

TSS Assurance Activities: The evaluator shall review the TSS section of the ST to determine that the TSS includes a description of the process of receiving sensitive data while the device is in a locked state. The evaluator shall also verify that the description indicates if sensitive data that may be received in the locked state is treated differently than sensitive data that cannot be received in the locked state. The description shall include the key scheme for encrypting and storing the received data, which must involve an asymmetric key and must prevent the sensitive data-at-rest from being decrypted by wiping all key material used to derive or encrypt the data (as described in the application note). The introduction to this section provides two different schemes that meet the requirements, but other solutions may address this requirement.

Section 6.3 of the ST describes how data is protected in a locked state. Section 2.2 of the KMD provides a discussion of the key scheme for encrypting and storing the received data. The discussion involves an asymmetric key and prevents sensitive data-at-rest from being decrypted by wiping key material.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform the tests in FCS_CKM_EXT.4 for all key material no longer needed while in the locked state and shall ensure that keys for the asymmetric scheme are addressed in the tests performed when transitioning to the locked state.

Test – The evaluator created a Knox container and added some email within the container to cause sensitive data keys to be used and logged those keys. The evaluator then locked the device so that sensitive data is encrypted and keys are no longer required. The evaluator then dumped the memory of the TOE device and searched for the keys logged. No keys were found. This test was performed with FCS_CKM_EXT.4.

2.3.5.3 MDFPP33:FDP_DAR_EXT.2.3

TSS Assurance Activities: The evaluator shall verify that the key hierarchy section of the TSS required for FCS_STG_EXT.2.1 includes the symmetric encryption keys (DEKs) used to encrypt sensitive data. The evaluator shall ensure that these DEKs are encrypted by a key encrypted with (or chain to a KEK encrypted with) the REK and password-derived or biometric-unlocked KEK.

The evaluator shall verify that the TSS section of the ST that describes the asymmetric key scheme includes the protection of any private keys of the asymmetric pairs. The evaluator shall ensure that any private keys that are not wiped and are stored by the TSF are stored encrypted by a key encrypted with (or chain to a KEK encrypted with) the REK and password-derived or biometric-unlocked KEK.

The evaluator shall also ensure that the documentation of the product's encryption key management is detailed enough that, after reading, the product's key management hierarchy is clear and that it meets the requirements to ensure the keys are adequately protected. The evaluator shall ensure that the documentation includes both an essay and one or more diagrams. Note that this may also be documented as separate proprietary evidence rather than being included in the TSS.

Section 6.2 of the ST discusses the key hierarchy at a high level and explains how keys are protected. Section 2.1 of the KMD provides a key hierarchy diagram and more detailed information about each key.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.4 MDFPP33:FDP_DAR_EXT.2.4

TSS Assurance Activities: The evaluator shall verify that the TSS section of the ST that describes the asymmetric key scheme includes a description of the actions taken by the TSF for the purposes of DAR upon transitioning to the unlocked state. These actions shall minimally include decrypting all received data using the asymmetric key scheme and re-encrypting with the symmetric key scheme used to store data while the device is unlocked.

Section 6.2 of the ST discusses the key hierarchy at a high level and explains how keys are protected. Section 2.1 of the KMD provides a key hierarchy diagram and more detailed information about each key.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.6 Subset Information Flow Control (MDFPP33:FDP_IFC_EXT.1)

2.3.6.1 MDFPP33:FDP_IFC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS section of the ST describes the routing of IP traffic through processes on the TSF when a VPN client is enabled. The evaluator shall ensure that the description indicates which traffic does not go through the VPN and which traffic does. The evaluator shall verify that a configuration exists for each baseband protocol in which only the traffic identified by the ST author as necessary for establishing the VPN connection (IKE traffic and perhaps HTTPS or DNS traffic) or needed for the correct functioning of the TOE is not encapsulated by the VPN protocol (IPsec). The evaluator shall verify that the TSS section describes any differences in the routing of IP traffic when using any supported baseband protocols (e.g. Wi-Fi or, LTE).

Section 6.3 of the ST states the TOE supports the installation of VPN Client applications, which can make use of the provided VPN APIs in order to configure the TOE's routing functionality to direct all traffic through the VPN. The TOE also includes an IPsec VPN Client that ensures all traffic other than traffic necessary to establish the VPN connection (for example, ARP, 802.11-2012 traffic, and IKEv2) flows through the VPN. The TOE routes all packets through the kernel's IPsec interface (ipsec0) when the VPN is active. The kernel compares packets routed through this interface to the SPDs configured for the VPN to determine whether to PROTECT, BYPASS, or DISCARD each packet. The vendor developed the TOE's VPN, when operating in CC Mode, to allow no configuration and always force all traffic through the VPN. The TOE ignores any IKEv2 traffic selector negotiations with the VPN GW and will always create an SPD PROTECT rule that matches all traffic. Thus, the kernel will match all packets, subsequently encrypt those packets, and finally forward them to the VPN Gateway.

Component Guidance Assurance Activities: The evaluator shall verify that one (or more) of the following options is addressed by the documentation:

- The description above indicates that if a VPN client is enabled, all configurations route all Data Plane traffic through the tunnel interface established by the VPN client.

- The AGD guidance describes how the user or administrator can configure the TSF to meet this requirement.

- The API documentation includes a security function that allows a VPN client to specify this routing.

Section 6.3 of the ST states the TOE also includes an IPsec VPN Client that ensures all traffic other than traffic necessary to establish the VPN connection (for example, ARP, 802.11-2012 traffic, and IKEv2) flows through the VPN. The TOE routes all packets through the kernel's IPsec interface (ipsec0) when the VPN is active. Section 6.5 of the Admin Guide provides the API for third party VPNs to use when directing all traffic to the VPN. Section 4.6 of the Admin Guide explains how to use the always on VPN and standard VPN tunnels.

Component Testing Assurance Activities: Test 38: If the ST author identifies any differences in the routing between Wi-Fi and cellular protocols, the evaluator shall repeat this test with a base station implementing one of the identified cellular protocols.

Step 1: The evaluator shall enable a Wi-Fi configuration as described in the AGD guidance (as required by FTP_ITC_EXT.1). The evaluator shall use a packet sniffing tool between the wireless access point and an Internet-connected network. The evaluator shall turn on the sniffing tool and perform actions with the device such as navigating to websites, using provided applications, and accessing other Internet resources. The evaluator shall verify that the sniffing tool captures the traffic generated by these actions, turn off the sniffing tool, and save the session data.

Step 2: The evaluator shall configure an IPsec VPN client that supports the routing specified in this requirement, and if necessary, configure the device to perform the routing specified as described in the AGD guidance. The evaluator shall ensure the test network is capable of sending any traffic identified as exceptions. The evaluator shall turn on the sniffing tool, establish the VPN connection, and perform the same actions with the device as performed in the first step, as well as ensuring that all exception traffic is generated. The evaluator shall verify that the sniffing tool captures traffic generated by these actions, turn off the sniffing tool, and save the session data.

Step 3: The evaluator shall examine the traffic from both step one and step two to verify that all Data Plane traffic is encapsulated by IPsec, modulo the exceptions identified in the SFR (if applicable). For each exception listed in the SFR, the evaluator shall verify that that traffic is allowed outside of the VPN tunnel. The evaluator shall examine the Security Parameter Index (SPI) value present in the encapsulated packets captured in Step two from the TOE to the Gateway and shall verify this value is the same for all actions used to generate traffic through the VPN. Note that it is expected that the SPI value for packets from the Gateway to the TOE is different than the SPI value for packets from the TOE to the Gateway. The evaluator shall be aware that IP traffic on the cellular baseband outside of the IPsec tunnel may be emanating from the baseband processor and shall verify with the manufacturer that any identified traffic is not emanating from the application processor.

Step 4: (Conditional: If ICMP is not listed as part of the IP traffic needed for the correct functioning of the TOE) The evaluator shall perform an ICMP echo from the TOE to the IP address of another device on the local wireless network and shall verify that no packets are sent using the sniffing tool. The evaluator shall attempt to send packets to the TOE outside the VPN tunnel (i.e. not through the VPN gateway), including from the local wireless network, and shall verify that the TOE discards them.

This test was performed in conjunction with the VPN client testing. The evaluator configured the test environment to send pings to all TOE devices from multiple hosts on the LAN and also to have each TOE device pinging multiple LAN hosts. The evaluator also browsed web pages on the Internet and on the LAN.

The results for FCS_IPSEC_EXT.1 test case 1 show pings originating from multiple addresses to the TOE devices that are successful, then successful VPN client connections (at which time the pings stop having responses), and eventually disconnects and resumes ping response from the TOE devices. Note the pings directed at the TOE devices are driven by the test scripts. Once the VPN client connections are made, the only traffic to/from the TOE devices are ESP packets and unanswered pings directed at the TOE devices.

2.3.7 Subset Information Flow Control (VPNC24:FDP_IFC_EXT.1)

2.3.7.1 VPNC24:FDP_IFC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS section of the ST describes the routing of IP traffic through processes on the TSF when a VPN client is enabled. The evaluator shall ensure that the description indicates which traffic does not go through the VPN and which traffic does and that a configuration exists for each baseband protocol in which only the traffic identified by the ST author as necessary for establishing the VPN connection (IKE traffic and perhaps HTTPS or DNS traffic) is not encapsulated by the VPN protocol (IPsec). The evaluator shall verify that the TSS section describes any differences in the routing of IP traffic when using any supported baseband protocols (e.g. Wi-Fi or, LTE).

See MDFPP33:FDP_IFC_EXT.1 for a description.

Component Guidance Assurance Activities: The evaluator shall verify that one (or more) of the following options is addressed by the documentation:

- The description above indicates that if a VPN client is enabled, all configurations route all Data Plane traffic through the tunnel interface established by the VPN client.

- The AGD guidance describes how the user and/or administrator can configure the TSF to meet this requirement.

- The API documentation includes a security function that allows a VPN client to specify this routing.

See MDFPP33:FDP_IFC_EXT.1 for a description.

Component Testing Assurance Activities: Refer to the EAs for FDP_IFC_EXT.1 in the MDF PP (included below). The only change to this SFR is that some selections are mandated, therefore the corresponding testing is mandatory. The actual testing for those selections is not changed.

Test 1: If the ST author identifies any differences in the routing between Wi-Fi and cellular protocols, the evaluator shall repeat this test with a base station implementing one of the identified cellular protocols.

Step 1: The evaluator shall enable a Wi-Fi configuration as described in the AGD guidance (as required by FTP_ITC_EXT.1). The evaluator shall use a packet sniffing tool between the wireless access point and an Internet-connected network. The evaluator shall turn on the sniffing tool and perform actions with the device such as navigating to websites, using provided applications, and accessing other Internet resources. The evaluator shall verify that the sniffing tool captures the traffic generated by these actions, turn off the sniffing tool, and save the session data.

Step 2: The evaluator shall configure an IPsec VPN client that supports the routing specified in this requirement, and if necessary, configure the device to perform the routing specified as described in the AGD guidance. The evaluator shall turn on the sniffing tool, establish the VPN connection, and perform the same actions with the device as performed in the first step. The evaluator shall verify that the sniffing tool captures traffic generated by these actions, turn off the sniffing tool, and save the session data.

Step 3: The evaluator shall examine the traffic from both step one and step two to verify that all Data Plane traffic is encapsulated by IPsec. The evaluator shall examine the Security Parameter Index (SPI) value present in the encapsulated packets captured in Step two from the TOE to the Gateway and shall verify this value is the same for all actions used to generate traffic through the VPN. Note that it is expected that the SPI value for packets from the Gateway to the TOE is different than the SPI value for packets from the TOE to the Gateway. The evaluator shall be aware that IP traffic on the cellular baseband outside of the IPsec tunnel may be emanating from the baseband processor and shall verify with the manufacturer that any identified traffic is not emanating from the application processor.

Step 4: The evaluator shall perform an ICMP echo from the TOE to the IP address of another device on the local wireless network and shall verify that no packets are sent using the sniffing tool. The evaluator shall attempt to send packets to the TOE outside the VPN tunnel (i.e. not through the VPN gateway), including from the local wireless network, and shall verify that the TOE discards them.

See MDFPP33:FDP_IFC_EXT.1 for a description.

2.3.8 Full Residual Information Protection (VPNC24:FDP_RIP.2)

2.3.8.1 VPNC24:FDP_RIP.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Requirement met by the platform

The evaluator shall examine the TSS to verify that it describes (for each supported platform) the extent to which the client processes network packets and addresses the FDP_RIP.2 requirement.

Requirement met by the TOE

'Resources' in the context of this requirement are network packets being sent through (as opposed to 'to', as is the case when a security administrator connects to the TOE) the TOE. The concern is that once a network packet is sent, the buffer or memory area used by the packet still contains data from that packet, and that if that buffer is re-used, those data might remain and make their way into a new packet. The evaluator shall check to ensure that the TSS describes packet processing to the extent that they can determine that no data will be reused when processing network packets. The evaluator shall ensure that this description at a minimum describes how the previous data are zeroized/overwritten, and at what point in the buffer processing this occurs.

Section 6.3 of the ST states the TOE has been designed to ensure that no residual information exists in network packets when the VPN is turned on. When the TOE allocates a new buffer for either an incoming or outgoing network packet, the new packet data will be used to overwrite any previous data in the buffer. If an allocated buffer exceeds the size of the packet, any additional space will be overwritten (padded) with zeros before the packet is forwarded (to the external network or delivered to the appropriate internal application).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.9 User Data Storage (MDFPP33:FDP_STG_EXT.1)

2.3.9.1 MDFPP33:FDP_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes the Trust Anchor Database implemented that contain certificates used to meet the requirements of this PP. This description shall contain information pertaining to how certificates are loaded into the store, and how the store is protected from unauthorized access (for example, UNIX permissions) in accordance with the permissions established in FMT_SMF.1 and FMT_MOF_EXT.1.1.

Section 6.3 of the ST states the TOE's Trusted Anchor Database consists of the built-in certificates (individually stored in `/system/etc/security/cacerts`) and any additional user or admin/MDM loaded certificates. The user can disable the built-in certificates or add new certificates using the TOE's Android user interface [Settings->Security-> Trusted Credentials]. The admin is able to load new certificates using the MDM. Disabled default certificates and user added certificates reside in the `/data/misc/user/0/cacerts-removed` and `/data/misc/user/0/cacerts-added` directories respectively. The built-in ones are protected, as they are part of the TSF's read only system partition, while the TOE protects user-loaded certificates by storing them with appropriate permissions to prevent modification by mobile applications. The TOE also stores the user-loaded certificates in the user's keystore.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.10 Inter-TSF User Data Transfer Protection (Applications) (MDFPP33:FDP_UPC_EXT.1/APPS)

2.3.10.1 MDFPP33:FDP_UPC_EXT.1.1/APPS

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.10.2 MDFPP33:FDP_UPC_EXT.1.2/APPS

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it describes that all protocols listed in the TSS are specified and included in the requirements in the ST.

protocols listed in the TSS are specified and included in the requirements in the ST.

Section 6.3 of the ST states the TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using IPsec, TLS, and HTTPS. The vendor provides the NIAPSEC library for application developers to use for Hostname Checking, Revocation Checking, and TLS Ciphersuite restriction. Mobile applications can use the following Android APIs for IPsec, TLS and HTTPS respectively:

android.net.VpnService

https://developer.android.com/reference/android/net/VpnService.html

com.samsung.android.knox.net.vpn

https://seap.samsung.com/api-references/android/reference/com/samsung/android/knox/net/vpn/package-summary.html

javax.net.ssl.SSLContext:

https://developer.android.com/reference/javax/net/ssl/SSLSocket

Developers then need to swap SocketFactory for SecureSocketFactory, part of a private library provided by Google.

Developers can request this library by emailing: niapsec@google.com

Component Guidance Assurance Activities: The evaluator shall verify that the API documentation provided according to Section 5.2.2 Class ADV: Development includes the security functions (protection channel) described in these requirements, and verify that the APIs implemented to support this requirement include the appropriate settings/parameters so that the application can both provide and obtain the information needed to assure mutual identification of the endpoints of the communication as required by this component.

The evaluator shall confirm that the operational guidance contains instructions necessary for configuring the protocols selected for use by the applications.

Section 6 of the Admin Guide provides APIs for all application protocols – TLS, HTTPS, and IPsec. The evaluator ensured the references contained enough information for a developer to configure each protocol – interfaces, parameters, etc. During the course of the evaluation, the evaluator had access to applications that used each protocol. The proper functioning of those applications supports the completeness of the API specification.

Component Testing Assurance Activities: Evaluation Activity Note: The following test requires the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

The evaluator shall write, or the developer shall provide access to, an application that requests protected channel services by the TSF. The evaluator shall verify that the results from the protected channel match the expected results according to the API documentation. This application may be used to assist in verifying the protected channel Evaluation Activities for the protocol requirements. The evaluator shall also perform the following tests:

Test 39: The evaluators shall ensure that the application is able to initiate communications with an external IT entity using each protocol specified in the requirement, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 40: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data are not sent in plaintext.

Test 39 – For TLS/HTTPS, see Test Case MDFPP33:FCS_TLSC_EXT.1.1 where an application is used to make TLS/HTTPS connections with all claimed ciphers. In each case an application data packet (showing the session is encrypted) is sent prior to disconnecting. For IPsec, see MDFPP33:FMT_MOF_EXT.1 test cases 70-3 where per-App VPN policies were configured demonstrating that VPNs could be assigned to specific applications.

Test 40 – This was performed with test case 39 where successful connections were made and the secure protocols were observed in the packet capture.

2.3.11 Inter-TSF User Data Transfer Protection (Bluetooth) (MDFPP33:FDP_UPC_EXT.1/BLUETOOTH)

2.3.11.1 MDFPP33:FDP_UPC_EXT.1.1/BLUETOOTH

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.2 MDFPP33:FDP_UPC_EXT.1.2/BLUETOOTH

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it describes that all protocols listed in the TSS are specified and included in the requirements in the ST.

Section 6.3 of the ST states the TOE provides APIs allowing non-TSF applications (mobile applications) the ability to establish a secure channel using Bluetooth BR/EDR and Bluetooth LE. This matches the associated requirement in the ST.

Component Guidance Assurance Activities: The evaluator shall verify that the API documentation provided according to Section 5.2.2 Class ADV: Development includes the security functions (protection channel) described in these requirements, and verify that the APIs implemented to support this requirement include the appropriate settings/parameters so that the application can both provide and obtain the information needed to assure mutual identification of the endpoints of the communication as required by this component.

The evaluator shall confirm that the operational guidance contains instructions necessary for configuring the protocols selected for use by the applications.

Section 6 of the Admin Guide provides APIs for the Bluetooth protocols – Bluetooth BR/EDR, and Bluetooth LE. The evaluator ensured the references contained enough information for a developer to configure each protocol – interfaces, parameters, etc. During the course of the evaluation, the evaluator had access to applications that used each protocol. The proper functioning of those applications supports the completeness of the API specification.

Component Testing Assurance Activities: Evaluation Activity Note: The following test requires the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

The evaluator shall write, or the developer shall provide access to, an application that requests protected channel services by the TSF. The evaluator shall verify that the results from the protected channel match the expected results according to the API documentation. This application may be used to assist in verifying the protected channel Evaluation Activities for the protocol requirements. The evaluator shall also perform the following tests:

Test 143: The evaluators shall ensure that the application is able to initiate communications with an external IT entity using each protocol specified in the requirement, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 144: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data are not sent in plaintext.

Test 143 – For Bluetooth BR/EDR, see FIA_BLT_EXT.2 test case 1. The Bluetooth BR/EDR packets were collected with a packet sniffer and examined. The evaluator concluded the packets are encrypted. The evaluator collected Bluetooth LE traffic as part of this test and concluded the packets are encrypted.

Test 144 – This was performed with test case 1 where successful connections were made and the secure protocols were observed in the packet capture. See Test Case FCS_CKM_EXT.8 test case 1 where the evaluator included captures of both successful and unsuccessful Bluetooth LE pairings.

2.3.12 Split Tunnel Prevention (VPNC24:FDP_VPN_EXT.1)

2.3.12.1 VPNC24:FDP_VPN_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS section of the ST describes the routing of IP traffic through processes on the TSF when a VPN client is enabled. The evaluator shall ensure that the description indicates which traffic does not go through the VPN and which traffic does and that a configuration exists for each baseband protocol in which only the traffic identified by the ST author is necessary for establishing the VPN connection (IKE traffic and perhaps HTTPS or DNS traffic) is not encapsulated by the VPN protocol (IPsec). The ST author shall also identify in the TSS section any differences in the routing of IP traffic when using any supported baseband protocols (e.g. Wi-Fi or LTE).

(TD0690 applied)

Section 6.3 of the ST explains the TOE supports the installation of VPN Client applications, which can make use of the provided VPN APIs in order to configure the TOE's routing functionality to direct all traffic through the VPN. The TOE also includes an IPsec VPN Client that ensures all traffic other than traffic necessary to establish the VPN connection (for example, ARP, 802.11-2012 traffic, and IKEv2) flows through the VPN. The TOE routes all packets through the kernel's IPsec interface (ipsec0) when the VPN is active. The kernel compares packets routed through this interface to the SPDs configured for the VPN to determine whether to PROTECT, BYPASS, or DISCARD each packet. The vendor developed the TOE's VPN, when operating in CC Mode, to allow no configuration and always force all traffic through the VPN. The TOE ignores any IKEv2 traffic selector negotiations with the VPN GW and will always create an SPD PROTECT rule that matches all traffic. Thus, the kernel will match all packets, subsequently encrypt those packets, and finally forward them to the VPN Gateway.

Component Guidance Assurance Activities: The evaluator shall verify that the following is addressed by the documentation:

- The description above indicates that if a VPN client is enabled, all configurations route all IP traffic (other than IP traffic required to establish the VPN connection) through the VPN client.

- The AGD guidance describes how the user and/or administrator can configure the TSF to meet this requirement.

(TD0690 applied)

Section 6.3 of the ST states the TOE also includes an IPsec VPN Client that ensures all traffic other than traffic necessary to establish the VPN connection (for example, ARP, 802.11-2012 traffic, and IKEv2) flows through the VPN. The TOE routes all packets through the kernel's IPsec interface (ipsec0) when the VPN is active. Section 6.5 of the Admin Guide provides the API for third party VPNs to use when directing all traffic to the VPN. Section 4.6 of the Admin Guide explains how to use the always on VPN and standard VPN tunnels.

Component Testing Assurance Activities: The evaluator shall perform the following test:

Step 1 - The evaluator shall use the platform to enable a network connection without using IPsec. The evaluator shall use a packet sniffing tool between the platform and an Internet-connected network. The evaluator shall turn on the sniffing tool and perform actions with the device such as navigating to websites, using provided applications, accessing other Internet resources (Use Case 1), accessing another VPN client (Use Case 2), or accessing an IPsec-capable network device (Use Case 3). The evaluator shall verify that the sniffing tool captures the traffic generated by these actions, turn off the sniffing tool, and save the session data.

Step 2 - The evaluator shall configure an IPsec VPN client that supports the routing specified in this requirement, and if necessary, configure the device to perform the routing specified as described in the AGD guidance. The evaluator shall turn on the sniffing tool, establish the VPN connection, and perform the same actions with the device as performed in the first step. The evaluator shall verify that the sniffing tool captures traffic generated by these actions, turn off the sniffing tool, and save the session data.

Step 3 - The evaluator shall examine the traffic from both step one and step two to verify that all IP traffic, aside from and after traffic necessary for establishing the VPN (such as IKE, DNS, and possibly HTTPS), is encapsulated by IPsec.

Step 4 - The evaluator shall attempt to send packets to the TOE outside the VPN connection and shall verify that the TOE discards them.

(TD0690 applied)

See MDFPP32:FDP_IFC_EXT.1 for test results where network activity is tested before and after establishing the VPN connection and the results show that once the VPN is established all traffic is flowing through the VPN connection.

2.4 Identification and authentication (FIA)

2.4.1 Authentication Failure Handling (MDFPP33:FIA_AFL_EXT.1)

2.4.1.1 MDFPP33:FIA_AFL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.1.2 MDFPP33:FIA_AFL_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.1.3 MDFPP33:FIA_AFL_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.1.4 MDFPP33:FIA_AFL_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.1.5 MDFPP33:FIA_AFL_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.1.6 MDFPP33:FIA_AFL_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS describes that a value corresponding to the number of unsuccessful authentication attempts since the last successful authentication is kept for each Authentication Factor interface. The evaluator shall ensure that this description also includes if and how this value is maintained when the TOE loses power, either through a graceful powered off or an ungraceful loss of power. The evaluator shall ensure that if the value is not maintained, the interface is after another interface in the boot sequence for which the value is maintained.

If the TOE supports multiple authentication mechanisms, the evaluator shall ensure that this description also includes how the unsuccessful authentication attempts for each mechanism selected in FIA_UAU.5.1 is handled. The evaluator shall verify that the TSS describes if each authentication mechanism utilizes its own counter or if multiple authentication mechanisms utilize a shared counter. If multiple authentication mechanisms utilize a shared counter, the evaluator shall verify that the TSS describes this interaction.

The evaluator shall confirm that the TSS describes how the process used to determine if the authentication attempt was successful. The evaluator shall ensure that the counter would be updated even if power to the device is cut immediately following notifying the TOE user if the authentication attempt was successful or not.

Section 6.4 of the ST explains the TOE maintains two separate lock screens: the device (Android) lock screen and a lock screen for the work profile. Each lock screen maintains individually stored (in separate Flash locations) failed login attempt counters.

The TOE maintains, for each lock screen, the number of failed logins since the last successful login, and upon reaching the maximum number of incorrect logins the TOE performs a full wipe of all protected data. For the device lock screen, this would mean a full wipe of all data on the device (a factory reset), while for the work profile lock screen this would remove all data associated with the work profile. The TOE maintains the number of failed logins across power-cycles (so for example, assuming a configured maximum retry of ten incorrect attempts, if one were to enter five incorrect passwords and power cycle the phone, the phone would only allow five more incorrect login attempts before wiping) by storing the number of logins remaining within its Flash file system. An administrator can adjust the number of failed logins to a value between one and 30 through an MDM.

For users with biometrics enabled, biometric authentication attempts are maintained along with the password attempts. In all cases, biometric or hybrid authentication mechanisms are non-critical and cannot be the authentication method that triggers an action (device or work profile wipe).

The maximum number of incorrect password authentication attempts can be configured to a value between 1 and 30. The maximum number of biometric attempts is 20 (this cannot be configured separately and this limit only applies to the device lock screen, not the work profile lock screen). The user can attempt 10 biometric attempts followed by the maximum number of password attempts before the TOE will wipe itself. For example, if the counter were set to 15, 10 biometric attempts would be followed by 15 password attempts before the device was wiped. Alternatively, the user might enter 14 incorrect passwords, and then 20 failed biometric authentication attempts followed by a final incorrect password attempt.

The TOE’s work profile provides its own lock screen, which allows password authentication or hybrid authentication (biometric and password). The hybrid authentication method requires the user to authenticate with a biometric and a password in sequential order. To login, the user must first enter his or her Knox biometric and only upon successfully verifying the user’s biometric, the TOE prompts the User for their Knox password, and the user must enter their password. Knox will count the number of incorrect passwords attempted, and wipe the work profile (and its associated data) after the user reaches the configured number of incorrect attempts.

The TOE validates passwords by providing them to Android’s Gatekeeper (which runs in the Trusted Execution Environment), and if the presented password fails to validate, the TOE increments the failed attempt counter before displaying a visual error to the user. The TOE validates biometric attempts through the biometric service (which runs in the Trusted Execution Environment), and if the presented biometric does not match the registered templates, the TOE increments the failed attempt counter before displaying a visual error to the user.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance describes how the administrator configures the maximum number of unique unsuccessful authentication attempts.

Section 3.3 of the Admin Guide contains a spreadsheet that provides the MDM command for setting the maximum number of failed login attempts, setMaximumFailedPasswordsForWipe. The device is wiped after the maximum failed attempts value is reached so no further configuration is necessary. The Knox container uses the same command for setting the maximum number of failed login attempts. When set in a KnoxConfigurationType, the setMaximumFailedPasswordsForDeviceDisable or the setMaximumFailedPasswordsForWipe settings will disable or wipe the Workspace, not the whole device.

Component Testing Assurance Activities: Test 41: The evaluator shall configure the device with all authentication mechanisms selected in FIA_UAU.5.1. The evaluator shall perform the following tests for each available authentication interface:

Test 41a: The evaluator shall configure the TOE, according to the AGD guidance, with a maximum number of unsuccessful authentication attempts. The evaluator shall enter the locked state and enter incorrect passwords until the wipe occurs. The evaluator shall verify that the number of password entries corresponds to the configured maximum and that the wipe is implemented.

Test 41b: [conditional] If the TOE supports multiple authentication mechanisms the previous test shall be repeated using a combination of authentication mechanisms confirming that the critical authentication mechanisms will cause the device to wipe and that when the maximum number of unsuccessful authentication attempts for a non-critical authentication mechanism is exceeded, the device limits authentication attempts to other available authentication mechanisms. If multiple authentication mechanisms utilize a shared counter, then the evaluator shall verify that the maximum number of unsuccessful authentication attempts can be reached by using each individual authentication mechanism and a combination of all authentication mechanisms that share the counter.

Test 42: The evaluator shall repeat test one, but shall power off (by removing the battery, if possible) the TOE between unsuccessful authentication attempts. The evaluator shall verify that the total number of unsuccessful authentication attempts for each authentication mechanism corresponds to the configured maximum and that the critical authentication mechanisms cause the device to wipe. Alternatively, if the number of authentication failures is not maintained for the interface under test, the evaluator shall verify that upon booting the TOE between unsuccessful authentication attempts another authentication factor interface is presented before the interface under test.

Test 41a – The evaluator configured the retry limit to 15 in order to test a maximum value and then attempted that number of failed attempts using a password (15 attempts) to ensure that they were each enforced. After 8 unsuccessful attempts the evaluator restarted the device using the adb shell (i.e., shutdown the device and then turned it back on - note that the battery in the Galaxy XCover6 Pro and Galaxy XCover Pro devices were removed, but the others have batteries that cannot be removed without disassembling the TOE). The restart kept the number of unsuccessful attempts and did not reset the count of attempts before wipe. After 15 failed attempts, the devices wiped themselves.

> Knox – The evaluator repeated the test for Knox. The evaluator set the maximum number of authentication failures for the Knox containers to 15. The evaluator then attempted to unlock locked Knox containers on all devices. The evaluator entered 5 bad passwords and then restarted the TOE (i.e., shutdown the device and then turned it back on – note that the battery in the Galaxy XCover6 Pro and Galaxy XCover Pro devices were removed, but the others have batteries that cannot be removed without disassembling the TOE). The evaluator then entered bad passwords until the failed count was exceeded and the Knox contain wiped.

Test 41b – The evaluator enrolled fingerprints and enabled a maximum of 10 authentication failures before wiping on each of the TOE devices. The evaluator configured fingerprints on all three TOE devices. During testing the evaluator entered a mixture of bad fingerprint and password attempts to hit both the biometric limit and critical password limit leading to a wipe.

> Knox - The evaluator configured fingerprint hybrid authentication on each TOE device and locked the container. The evaluator enabled a maximum of 10 authentication failures before wiping the container. The evaluator then failed 25 biometric attempts at which point, the device switched to passwords. The evaluator failed 10 password attempts and the container wiped.

Test 42 - The evaluator performed this as part of test 1a where a reboot is performed on each TOE device in the middle of that test case. The battery was removed from the Galaxy XCover6 Pro devices as part of the reboot. The other devices do not have removable batteries.

Knox – The evaluator performed this as part of test 41a where the TOE devices were rebooted in the middle of the testing to ensure the counters continued without disruption.

2.4.2 Bluetooth User Authorization (BT10:FIA_BLT_EXT.1)

2.4.2.1 BT10:FIA_BLT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it contains a description of when user permission is required for Bluetooth pairing, and that this description mandates explicit user authorization via manual input for all Bluetooth pairing, including application use of the Bluetooth trusted channel and situations where temporary (non-bonded) connections are formed.

Section 6.4 of the ST states the TOE requires explicit user authorization before it will pair with a remote Bluetooth device. When pairing with another device, the TOE requires that the user either confirm that a displayed numeric passcode matches between the two devices or that the user enter (or choose) a numeric passcode that the peer device generates (or must enter).

Component Guidance Assurance Activities: The evaluator shall examine the API documentation provided according to Section 5.2.2 and verify that this API documentation does not include any API for programmatic entering of pairing information (e.g. PINs, numeric codes, or 'yes/no' responses) intended to bypass manual user input during pairing. The evaluator shall examine the AGD guidance to verify that these user authorization screens are clearly identified and instructions are given for authorizing Bluetooth pairings.

Section 4.3 of the Admin Guide provides a reference to user documentation for Bluetooth functions, including pairing devices. The Admin Guide provides the steps for pairing a Bluetooth device. It then provides a reference in case the user wants to see pictures.

The evaluator examined the API information documented in the Admin Guide and did not find any API included for programmatic entering of pairing information intended to bypass manual user input during pairing.

Component Testing Assurance Activities: Test 1: The evaluator shall perform the following steps:

Step 1: Initiate pairing with the TOE from a remote Bluetooth device that requests no man-in-the-middle protection, no bonding, and claims to have NoInputNoOutput input-output (IO) capability. (Such a device will attempt to evoke behavior from the TOE that represents the minimal level of user interaction that the TOE supports during pairing.)

Step 2: Verify that the TOE does not permit any Bluetooth pairing without explicit authorization from the user (e.g. the user must have to minimally answer 'yes' or 'allow' in a prompt).

Test 1 – The evaluator requested Bluetooth pairing with a device that requests no man-in-the-middle protection, no bonding, and claims to have NoInputNoOutput input-output (IO) capability. The evaluator observed that the user had to explicitly authorize the pairing. In the test case, the user denied the request and the devices were not paired.

2.4.3 Bluetooth Mutual Authentication (BT10:FIA_BLT_EXT.2)

2.4.3.1 BT10:FIA_BLT_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS describes how data transfer of any type is prevented before the Bluetooth pairing is completed. The TSS shall specifically call out any supported RFCOMM and L2CAP data transfer mechanisms. The evaluator shall ensure that the data transfers are only completed after the Bluetooth devices are paired and mutually authenticated.

Section 6.4 of the ST states the TOE requires explicit user authorization or user authorization before data transfers over the link. When transferring data with another device, the TOE requires that the user must confirm an authorization popup displayed allowing data transfer (Obex Object Push) or confirm the user passkey displayed numeric passcode matches between the two devices (RFCOMM).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following test:

Test 1: The evaluator shall use a Bluetooth tool to attempt to access TOE files using the OBEX Object Push service and verify that pairing and mutual authentication are required by the TOE before allowing access. (If the OBEX Object Push service is unsupported on the TOE, a different service that transfers data over Bluetooth L2CAP and/or RFCOMM may be used in this test.)

Test 1: The evaluator turned on Bluetooth sniffing on a device and then paired the TOE with a second device and transferred a file. A packet capture confirmed mutual authentication.

2.4.4 Rejection of Duplicate Bluetooth Connections (BT10:FIA_BLT_EXT.3)

2.4.4.1 BT10:FIA_BLT_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS describes how Bluetooth connections are maintained such that two devices with the same Bluetooth device address are not simultaneously connected and such that the initial session is not superseded by any following session initialization attempts.

Section 6.4 of the ST explains the TOE tracks active connections and actively ignores connection attempts from Bluetooth device addresses for which the TOE already has an active connection.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following steps:

Step 1: Pair the TOE with a remote Bluetooth device (DEV1) with a known address BD_ADDR. Establish an active session between the TOE and DEV1 with the known address BD_ADDR.

Step 2: Attempt to pair a second remote Bluetooth device (DEV2) claiming to have a Bluetooth device address matching DEV1 BD_ADDR to the TOE. Using a Bluetooth protocol analyzer, verify that the pairing attempt by DEV2 is not completed by the TOE and that the active session to DEV1 is unaffected.

Step 3: Attempt to initialize a session to the TOE from DEV2 containing address DEV1 BD_ADDR. Using a Bluetooth protocol analyzer, verify that the session initialization attempt by DEV2 is ignored by the TOE and that the initial session to DEV1 is unaffected.

Test 1 - The evaluator configured two remote devices to share the same Bluetooth address. The evaluator then created an active session between the TOE and the paired device by enabling Bluetooth tethering on the paired peer. Finally, the evaluator attempted to pair the other cloned device to the TOE to observe that failed and did not interrupt the active connection. The evaluator analyzed the packet capture and observed the TOE connects with the original peer device and analogous packets can be seen in the original peer packet capture and there are many packets associated with the Internet sharing connection. The evaluator examined the clone device packet capture but found evidence of only limited packets exchanged and no communication channels.

2.4.5 Secure Simple Pairing (BT10:FIA_BLT_EXT.4)

2.4.5.1 BT10:FIA_BLT_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.5.2 BT10:FIA_BLT_EXT.4.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the secure simple pairing process.

Section 6.4 of the ST states the TOE’s Bluetooth host and controller support Bluetooth Secure Simple Pairing and the TOE utilizes this pairing method when the remote host also supports it.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Test 1: The evaluator shall perform the following steps:

Step 1: Initiate pairing with the TOE from a remote Bluetooth device that supports Secure Simple Pairing.

Step 2: During the pairing process, observe the packets in a Bluetooth protocol analyzer and verify that the TOE claims support for both 'Secure Simple Pairing (Host Support)' and 'Secure Simple Pairing (Controller Support)' during the LMP Features Exchange.

Step 3: Verify that Secure Simple Pairing is used during the pairing process.

Test 1 – The evaluator paired two devices. The evaluator was able to analyze the packet capture and saw where the TOE set flags to indicate that it supported Secure Simple Pairing. However, while the flag does not differentiate host vs. controller, given the TOE pairings where in each case one is a host and one the controller it can be concluded that both host and controller are supported.

2.4.6 Trusted Bluetooth Device User Authorization (BT10:FIA_BLT_EXT.6)

2.4.6.1 BT10:FIA_BLT_EXT.6.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes all Bluetooth profiles and associated services for which explicit user authorization is required before a remote device can gain access. The evaluator shall also verify that the TSS describes any difference in behavior based on whether or not the device has a trusted relationship with the TOE for that service (i.e. whether there are any services that require explicit user authorization for untrusted devices that do not require such authorization for trusted devices). The evaluator shall also verify that the TSS describes the method by which a device can become 'trusted'.

Section 6.4 of the ST state the TOE requires that OPP and MAP profile connections have explicit user authorization before granting access to trusted remote devices on a per service basis (as opposed to a per app basis). The TOE requires that untrusted devices have explicit user authorization before granting access to untrusted remote devices for the OPP Bluetooth profile on a per service basis (as opposed to a per app basis).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following tests for each service protected according to this requirement:

Test 1: While the service is in active use by an application on the TOE, the evaluator shall attempt to gain access to a 'protected' Bluetooth service (as specified in the assignment in FIA_BLT_EXT.6.1) from a 'trusted' remote device. The evaluator shall verify that the user is explicitly asked for authorization by the TOE to allow access to the service for the particular remote device. The evaluator shall deny the authorization on the TOE and verify that the remote attempt to access the service fails due to lack of authorization.

Test 2: The evaluator shall repeat Test 1, this time allowing the authorization and verifying that the remote device successfully accesses the service.

Test 1 - Without pairing the test devices, the evaluator used a third-party tool with Bluetooth enabled on the TOE devices, to attempt to connect using the MAP profile. When prompted the tester denied the connection and observed the connection failed and MAP access was not allowed. This test was repeated with OPP.

Test 2 - Without pairing the test devices, the evaluator used a third-party tool with Bluetooth enabled on the TOE devices, to attempt to connect using the MAP profile. When prompted the tester accepted the connection and observed the connection succeed and MAP access was allowed. This test was repeated with OPP.

2.4.7 Untrusted Bluetooth Device User Authorization (BT10:FIA_BLT_EXT.7)

2.4.7.1 BT10:FIA_BLT_EXT.7.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The TSS evaluation activities for this component are addressed by FIA_BLT_EXT.6.

See BT10:FIA_BLT_EXT.6.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following tests if the TSF differentiates between 'trusted' and 'untrusted' devices for the purpose of granting access to services. If it does not, then the test evaluation activities for FIA_BLT_EXT.6 are sufficient to satisfy this component.

Test 1: While the service is in active use by an application on the TOE, the evaluator shall attempt to gain access to a 'protected' Bluetooth service (as specified in the assignment in FIA_BLT_EXT.7.1) from an 'untrusted' remote device. The evaluator shall verify that the user is explicitly asked for authorization by the TOE to allow access to the service for the particular remote device. The evaluator shall deny the authorization on the TOE and verify that the remote attempt to access the service fails due to lack of authorization.

Test 2: The evaluator shall repeat Test 1, this time allowing the authorization and verifying that the remote device successfully accesses the service.

Test 3: (conditional): If there exist any services that require explicit user authorization for access by untrusted devices but not by trusted devices (i.e. a service that is listed in FIA_BLT_EXT.7.1 but not FIA_BLT_EXT.6.1), the evaluator shall repeat Test 1 for these services and observe that the results are identical. That is, the evaluator shall use these results to verify that explicit user approval is required for an untrusted device to access these services, and failure to grant this approval will result in the device being unable to access them.

Test 4: (conditional): If test 3 applies, the evaluator shall repeat Test 2 using any services chosen in Test 3 and observe that the results are identical. That is, the evaluator shall use these results to verify that explicit user approval is required for an untrusted device to access these services, and granting this approval will result in the device being able to access them.

Test 5: (conditional): If test 3 applies, the evaluator shall repeat Test 3 except this time designating the device as 'trusted' prior to attempting to access the service. The evaluator shall verify that access to the service is granted without explicit user authorization (because the device is now trusted and therefore FIA_BLT_EXT.7.1 no longer applies to it). That is, the evaluator shall use these results to demonstrate that the TSF will grant a device access to different services depending on whether or not the device is trusted.

Test 1 – The evaluator enabled Bluetooth on two unpaired devices and then attempted a Bluetooth OPP file transfer to the TOE from the other device. The evaluator verified that the user was explicitly asked for authorization by the TOE to allow access to the service for the particular remote device. The evaluator declined the request and confirmed that the file was not transferred due to lack of authorization.

Test 2 – The evaluator repeated Test 1, this time allowing the authorization and verifying that the connection was accepted and file was successfully transferred for OPP.

Test 3 – Not applicable – the same authorization is required for profile access regardless of trusted or not trusted.

Test 4 -. Not applicable – test 3 does not apply.

Test 5 - Not applicable – test 3 does not apply.

2.4.8 Biometric enrolment - per TD0714 (BIO11:FIA_MBE_EXT.1)

2.4.8.1 BIO11:FIA_MBE_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to understand how the TOE enrols a user.

Section 6.4 of the ST states the TOE provides a mechanism to enroll user biometrics for authentication. The enrollment process requires the user to have a password set and to be authenticated successfully prior to being able to start the enrollment process.

Component Guidance Assurance Activities: The evaluator shall examine the AGD guidance to confirm that a user is required to enter one's valid NBAF before the biometric enrolment.

Section 2.5.1 of the Admin Guide explains that users must first authenticate using their password before gaining access to biometric authentication.

Component Testing Assurance Activities: The evaluator shall perform the following steps to verify that the TOE performs the biometric enrolment correctly.

1. The evaluator shall try to enrol without setting a NBAF and confirm that it is not possible to enrol.

2. The evaluator shall set a NBAF and confirm that enrolment is not possible without entering the NBAF correctly beforehand.

See Test Case FIA_UAU.6/CREDENTIAL test case 47 where the evaluator showed that a NBAF (password) must first be configured in order to enroll a BAF (fingerprint). Additionally, within the same results, the evaluator also confirmed that even with a NBAF (password) configured, a BAF (fingerprint) enrollment is not possible without entering the NBAF (password) correctly beforehand.

2.4.9 Quality of biometric templates for biometric enrolment (BIO11:FIA_MBE_EXT.2)

2.4.9.1 BIO11:FIA_MBE_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS and BMD to understand how the TOE generates templates of sufficient quality from samples at enrolment.

Section 6.4 of the ST states the TSF determines the quality of a sample before using it to enroll or verify a user. The fingerprint systems utilize different sensors to capture the data, but the data analysis of the quality is common. Fingerprint sample quality is determined using three measures – completeness, clarity, and minutia. The KMD explains how each measure is used in determining quality.

Component Guidance Assurance Activities: The evaluator shall also examine the AGD guidance about how the TOE supports a user to enrol correctly and how the TOE behaves when low quality samples are presented to the TOE for enrolment.

The evaluator shall examine the quality assessment criteria report to check the efficacy of quality metrics to confirm that the TOE can select enough quality of samples from which the TOE generates templates of sufficient quality.

Section 2.5.1 of the Admin Guide explains that when enrolling a fingerprint, the device provides on-screen guidance to capture quality data, asking users to move their finger to cover a wider area. To acquire optimal fingerprint samples, the device prompts users to ensure they maintain proper finger placement on the sensor, providing multiple user-friendly messages until quality data acquisition is complete.

Component Testing Assurance Activities: Enrolment templates

The evaluator shall perform the following test to verify that the TOE generates templates of sufficient quality.

The following test steps require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

1. The evaluator shall perform biometric enrolment that results in creation of samples from which templates will be created that do not satisfy the assessment criteria described in quality assessment criteria report. Methods to create low-quality samples are described in the report (e.g. varying temperature / humidity conditions of the finger skin, low physical pressure, too less presentation time or incorrect finger positioning angles for fingerprint verification)

2. The evaluator shall check the TOE internal data (e.g. quality scores and quality threshold) to confirm that the TOE does not create enrolment templates from samples that do not meet the assessment criteria specified in the quality assessment criteria report

3. The evaluator shall check the TOE internal data (e.g. quality scores and quality threshold) to confirm that any enrolment templates are created by TOE from samples that meet the assessment criteria specified in the quality assessment criteria report correctly

Authentication templates

The evaluator shall perform the following test to verify that the TOE generates authentication templates of sufficient quality only if the evaluator judges that creating authentication templates is feasible.

The following test steps require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

1. The evaluator shall enrol to the biometric system

2. The evaluator shall present biometric samples repeatedly to trigger the TOE to create authentication templates

3. The evaluator shall check the TOE internal data (e.g. quality scores and quality threshold) to confirm that the TOE does not create authentication templates from samples that do not meet the assessment criteria specified in the quality assessment criteria report

4. The evaluator shall check the TOE internal data (e.g. quality scores and quality threshold) to confirm that any authentication templates created by TOE from samples that meet the assessment criteria specified in the quality assessment criteria report correctly

Starting with a phone that had no configured biometric authentication factors enrolled, the evaluator attempted to enroll a fingerprint to be used as an authentication factor. This fingerprint is registered through the fingerprint scanner on the TOE device, analyzed to meet a series of quality criteria, and then reported back to the system if a biometric template could be created from the sample.

The evaluator first tried to enroll a fingerprint and performed several steps to ensure it was of poor quality. The TOE recorded failure logs for each sample that was not recorded. On bad samples, the UI would also update with occasional suggestions for how the user could generate a better sample. The evaluator verified the logs recorded the correct information. The evaluator then generated a good fingerprint sample and verified the logs. A good fingerprint has been shown oy be used in BIO11:FIA_MBV_EXT.1 that it could be used to unlock and authenticate the device.

2.4.10 Biometric verification (BIO11:FIA_MBV_EXT.1)

2.4.10.1 BIO11:FIA_MBV_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.10.2 BIO11:FIA_MBV_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS and BMD to understand how the TOE verifies a user with one's biometric characteristics.

Section 6.4 of the ST states The TOE’s fingerprint sensor provides FAR and FRR rates. For a password-only configuration, the SAFAR claim would be 1:1,000,000 when set for 10 attempts. The password minimum length is 4 characters and there are 93 possible characters that can be used in the password. This is not claimed since this configuration (i.e. no biometric authentication allowed) would not require FIA_BMG_EXT.1, but is shown here for the overall SAFAR calculations.

$$SAFAR_{password} = 1 - \left(1 - {}^{1}\!/_{93^{4}}\right)^{30} = 4.010 * 10^{-7}$$

The FAR for fingerprint is 1:10,000 and the FRR is 3%. The SAFAR when a fingerprint is in use is 1:1,000 based on a maximum of 10 attempts of the biometric as noted in the worst-case scenario.

$$SAFAR_{fingerprint} = 1 - (1 - 10^{-4})^{10} = 9.996 * 10^{-4}$$

For all SAFAR calculations the password is considered a critical factor for all combined factor SAFARany calculations as detailed in PP_MD_V3.2 section G.4. The values are accepted because they are within the 1% margin allowed by BIO11.

For devices which support fingerprint biometrics, SAFARfingerprint can be used for the calculation of a worst-case scenario.

$$SAFAR_{any-fingerprint} = 1 - \left(1 - SAFAR_{ff}\right) * (1 - SAFAR_{password})$$

$$SAFAR_{any-ff} = 1.00022 \ * 10^{-3}$$

The Knox Platform for Enterprise provides hybrid (multi-factor: password and biometric) authentication. When using the hybrid authentication mechanism, the user must enter a correct biometric factor and then a correct password in order to successfully unlock the work profile.

The SAFAR when a hybrid mechanism is in use is equal to SAFARpassword. The maximum number of biometric attempts when using hybrid is 50, but eventually the biometric factor must be entered correctly and then one must always enter a correct password. Given the potential maximum 50 biometric attempts, although that could take a very long time given some imposed delays, that factor is discounted in calculating the SAFAR. The password minimum length is 4 characters and there are 93 possible characters that can be used in the password.

$$SAFAR_{hybrid} = SAFAR_{password}$$

The biometric FAR/FRR values are tested internally by two independent groups using different methodologies. The first set of tests are “offline” in that a specially configured device is connected to a test harness and used to enroll biometric samples for storage. The test harness then uses the samples to run through numerous combinations of the samples to determine FAR/FRR results that are used to tune the algorithms controlling the biometric system. Once testing is complete a second set of tests are performed in an “online” manner where the testing is done with users directly testing on a live device.

All devices with different hardware combinations that could affect the biometric subsystem are tested. For example, both the Exynos and Qualcomm versions of the mobile devices are tested individually since the TrustZone components in each device are different. These tests are integrated into the production process and so are repeated continually during the development process.

Component Guidance Assurance Activities: The evaluator shall also examine the guidance about how the TOE supports a user to verify one's biometric correctly and how the TOE behaves when biometric verification is succeeded or failed.

The evaluator shall examine developer's performance report to verify that the developer conducts the objective and repeatable performance testing. Minimum requirements for conducting performance testing are defined in Chapter 8, Developer's performance report and its assessment strategy.

Requirements defined in Chapter 8, Developer's performance report and its assessment strategy are based on ISO/IEC 19795. This standard specifies requirements on performance test protocol, recording and reporting of results based on the best practices developed by relevant organizations. The evaluator shall confirm that developer's performance report meets all requirements in Chapter 8, Developer's performance report and its assessment strategy and seek a rationale if the developer's performance report does not meet any requirements and determine whether the rationale is valid or not.

Finally, the evaluator shall check that the estimated upper bound confidence interval of error rates (FRR/FAR or FNMR/FMR) reported in the developer's performance report do not exceed the claimed error rates specified in the FIA_MBV_EXT.1.2.

Section 4.1 of the Admin Guide explains when allowed, a user will be able to enroll fingerprint biometrics for use at the lock screen as an alternative to entering a password. Detailed instructions for configuring these methods can be found under the “Secure” or “Security” section of the guide for the specific device. In these instructions, users can see what happens with successful and failed attempts to use fingerprints.

Component Testing Assurance Activities: None Defined

2.4.11 Quality of biometric samples for biometric verification (BIO11:FIA_MBV_EXT.2)

2.4.11.1 BIO11:FIA_MBV_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS and BMD to understand how the TOE checks quality of samples captured.

See BIO11_FIA_MBV_EXT.1 for a summary. The detailed information is presented in the KMD.

Component Guidance Assurance Activities: The evaluator shall also examine the AGD guidance about how the TOE supports a user to verify correctly and how the TOE behaves when low quality samples are presented to the TOE for verification.

The evaluator shall examine the quality assessment criteria report to check the efficacy of quality metrics to confirm that the TOE can select enough quality of samples from which the TOE generates templates of sufficient quality.

Section 2.5.1 of the Admin Guide explains that when enrolling a fingerprint, the device provides on-screen guidance to capture quality data, asking users to move their finger to cover a wider area. To acquire optimal fingerprint samples, the device prompts users to ensure they maintain proper finger placement on the sensor, providing multiple user-friendly messages until quality data acquisition is complete.

Component Testing Assurance Activities: The evaluator shall perform the following test to verify that the TOE checks the quality of samples based on the assessment criteria.

The following test steps require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

1. The evaluator shall present biometric low-quality samples for biometric verification that do not satisfy the assessment criteria described in quality assessment criteria report. Methods to create low-quality samples are described in the report (e.g. varying temperature / humidity conditions of the finger skin, low physical pressure, too less presentation time or incorrect finger positioning angles for fingerprint verification)

2. The evaluator shall check the TOE internal data (e.g. quality scores and quality threshold) to confirm that the TOE rejects any samples that do not meet the assessment criteria specified in the quality assessment criteria report

3. The evaluator shall check the TOE internal data (e.g. quality scores and quality threshold) to confirm that any samples accepted by TOE meet the assessment criteria specified in the quality assessment criteria report correctly

The evaluator followed a similar analysis process as to what was performed for BIO11:FIA_MBE_EXT.2 which covers enrollment. Although this requirement deals with verification, the TOE relies on the same Biometric HAL messages to prove the accuracy of quality samples. The evaluator first started by enrolling their finger into the TOE's list of trusted biometric authentication factors in the settings application. With the fingerprint enrolled, the evaluator could attempt to use the fingerprint for authentication purposes, such as unlocking the phone. The evaluator locked the phone and then attempted to unlock it using the fingerprint scanner. Rather than use the correct fingerprint, the evaluator supplied a series of incorrect samples. The evaluator observed several errors when trying to do this until the TOE eventually locked out the fingerprint. At this point the evaluator had to use the password to unlock the phone. The evaluator again checked the logs following the analysis in BIO11:FIA_MBE_EXT.2 and found the relevant logs. The evaluator them performed a successful authentication using a fingerprint and found the associated logs.

2.4.12 Port Access Entity Authentication (WLANC10:FIA_PAE_EXT.1)

2.4.12.1 WLANC10:FIA_PAE_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall demonstrate that the TOE has no access to the test network. After successfully authenticating with an authentication server through a wireless access system, the evaluator shall demonstrate that the TOE does have access to the test network.

Test 2: The evaluator shall demonstrate that the TOE has no access to the test network. The evaluator shall attempt to authenticate using an invalid client certificate, such that the EAP-TLS negotiation fails. This should result in the TOE still being unable to access the test network.

Test 3: The evaluator shall demonstrate that the TOE has no access to the test network. The evaluator shall attempt to authenticate using an invalid authentication server certificate, such that the EAP-TLS negotiation fails. This should result in the TOE still being unable to access the test network.

Test 1 – The evaluator attempted to browse a web page to show a lack of connectivity, then connected to an access point using valid credentials, and then attempted to browse a web page to show successful connectivity.

Test 2 – The evaluator installed a client certificate that was invalid on the device. The connection was rejected because the client certificate was not valid. The evaluator attempted to browse a web page to show a lack of connectivity.

Test 3 – The evaluator installed a server certificate that was expired on the device. The connection was rejected because the server certificate was not valid. The evaluator attempted to browse a web page to show a lack of connectivity.

2.4.13 Password Management (MDFPP33:FIA_PMG_EXT.1)

2.4.13.1 MDFPP33:FIA_PMG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it provides guidance to security administrators on the composition of strong passwords, and that it provides instructions on setting the minimum password length. The evaluator shall also perform the following tests. Note that one or more of these tests can be performed with a single test case.

Section 4.1.1 of the Admin Guide provides instructions for selecting strong passwords by referencing the NIST publication, NIST SP 800-63B, section 5.1.1, Memorized Secrets. Section 3.3 of the Admin Guide provides a spreadsheet that contains the MDM command for setting the minimum password length, setPasswordMinimumLength.

KNOX - Section 3.3 of the Admin Guide provides the MDM command for setting the minimum password length for the container in its spreadsheet, setPasswordMinimumLength.

Component Testing Assurance Activities: Test 43: The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, rule characteristics, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

Test 43 – The evaluator set the minimum password length to 16 characters. The evaluator then went through all the possible characters claimed in the ST, 16 at a time. The evaluator was able to verify that all claimed characters were valid. The evaluator attempted to set a password of 15 characters and was denied as expected. The evaluator attempted to set a password greater than 16 and was denied when the max length was set to 16.

Knox – The evaluator repeated the same test on the Knox container and demonstrated that the minimum and maximum length of 16 functioned properly and all the claimed characters were valid.

2.4.14 Authentication Throttling (MDFPP33:FIA_TRT_EXT.1)

2.4.14.1 MDFPP33:FIA_TRT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the method by which authentication attempts are not able to be automated. The evaluator shall ensure that the TSS describes either how the TSF disables authentication via external interfaces (other than the ordinary user interface) or how authentication attempts are delayed in order to slow automated entry and shall ensure that this delay totals at least 500 milliseconds over 10 attempts for all authentication mechanisms selected in FIA_UAU.5.1.

Section 6.4 of the ST states the TOE allows users to authenticate through external ports (either a USB keyboard or a Bluetooth keyboard paired in advance of the login attempt). If not using an external keyboard, a user must authenticate through the standard User Interface (using the TOE touchscreen). The TOE limits the number of authentication attempts through the UI to no more than five attempts within 30 seconds (irrespective of what keyboard the operator uses). Thus if the current [the nth] and prior four authentication attempts have failed, and the n-4th attempt was less than 30 seconds ago, the TOE will prevent any further authentication attempts until 30 seconds has elapsed. Note as well that the TOE will wipe itself when it reaches the maximum number of unsuccessful authentication attempts.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4.15 Multiple Authentication Mechanisms (MDFPP33:FIA_UAU.5)

2.4.15.1 MDFPP33:FIA_UAU.5.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.15.2 MDFPP33:FIA_UAU.5.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS describes each mechanism provided to support user authentication and the rules describing how the authentication mechanisms provide authentication.

Specifically, for all authentication mechanisms specified in FIA_UAU.5.1, the evaluator shall ensure that the TSS describes the rules as to how each authentication mechanism is used. Example rules are how the authentication mechanism authenticates the user (i.e. how does the TSF verify that the correct password or biometric sample was entered), the result of a successful authentication (i.e. is the user input used to derive or unlock a key) and which authentication mechanism can be used at which authentication factor interfaces (i.e. if there are times, for example, after a reboot, that only specific authentication mechanisms can be used). If multiple BAFs are claimed in FIA_MBV_EXT.1.1 in the Biometric Enrollment and Verification, version 1.1, the interaction between the BAFs must be described. For example, whether the multiple BAFs can be enabled at the same time.

Section 6.4 of the ST explains the TOE allows the following authentication methods at the different lock screens in CC mode. The available biometrics are dependent on the lock screen being used, MDM configuration and enrolled templates. Table 18 (34 in the TSS) shows which authentication methods are available at each lock screen (this is Table 36 in the ST document).

Device Lock	Knox Workspace Container Lock
Password, Biometrics	Password, Hybrid

Table 18 – Allowed Lock Screen Authentication Methods

The TOE prohibits other authentication mechanisms such as pattern or swipe. Use of Smart Lock mechanisms (on-body detection, trusted places, trusted devices, trusted face, and trusted voice) and PIN can be blocked through management controls. Upon restart or power-up the user can only use a password for authentication to unlock the device. Once past this initial authentication, the user is able to use one of the configured methods at the device lock screen or the work profile lock screen.

Section 6.9 of the ST explains the work profile allows the user to authenticate using a password, or a hybrid method requiring both a biometric and the password at the same time. The TOE prohibits other authentication mechanisms, such as pattern, PIN, or biometric by themselves.

Component Guidance Assurance Activities: The evaluator shall verify that configuration guidance for each authentication mechanism is addressed in the AGD guidance.

Section 2.5.1 of the Admin Guide explains that when enrolling a fingerprint, the device provides on-screen guidance to capture quality data, asking users to move their finger to cover a wider area. To acquire optimal fingerprint samples, the device prompts users to ensure they maintain proper finger placement on the sensor, providing multiple user-friendly messages until quality data acquisition is complete. Section 4.1 states detailed instructions for configuring these methods can be found under the "Secure" or "Security" section of the guide for the specific device. However as explained in Section 2.5.1, the device provides on-screen help for enrolling a fingerprint. Section 4.1.1 discusses setting a password and 4.1.2 discusses setting two factor (aka hybrid) authentication factors. These discussions apply to the devices as a whole and the workspace.

Component Testing Assurance Activities: Test 44: For each authentication mechanism selected in FIA_UAU.5.1, the evaluator shall enable that mechanism and verify that it can be used to authenticate the user at the specified authentication factor interfaces.

Test 45: For each authentication mechanism rule, the evaluator shall ensure that the authentication mechanisms behave accordingly.

Test 44 – This was tested as part of FMT_MOF_EXT.1 test case 70, function 23 where the evaluator tested restrictions in regard to biometrics and tested that each biometric (fingerprint) could successfully be used to unlock the TOE. The evaluator also tested that a password could unlock the TOE and the vast majority of testing involved regular unlocking using a configured password.

> KNOX - This was tested as part of FMT_MOF_EXT.1 test case 70, function 23 (KNOX) where the evaluator tested restrictions in regard to hybrid authentication methods and tested that hybrid authentication could successfully be used – first a biometric factor followed by a password. The evaluator also tested that a password-only could unlock the container when not configured for hybrid (or hybrid was blocked) and the vast majority of testing involved regular unlocking of containers using a configured password.

Test 45 – This was tested as part of Test Cases FIA_UAU.5 test case 1 and FIA_AFL_EXT.1 where the authentication rules are tested. The evaluator also tested that biometric authentication is disabled upon reboot and only passwords are accepted.

2.4.16 Re-Authenticating (Credential Change) (MDFPP33:FIA_UAU.6/CREDENTIAL)

2.4.16.1 MDFPP33:FIA_UAU.6.1/CREDENTIAL

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Test 46: The evaluator shall configure the TSF to use the Password Authentication Factor according to the AGD guidance. The evaluator shall change Password Authentication Factor according to the AGD guidance and verify that the TSF requires the entry of the Password Authentication Factor before allowing the factor to be changed.

Test 47: [conditional] If biometric in accordance with the Biometric Enrollment and Verification, version 1.1 is selected in FIA_UAU.5.1, for each BAF claimed in FIA_MBV_EXT.1.1 in the Biometric Enrollment and Verification, version 1.1 the evaluator shall configure the TSF to use the BAF, which includes configuring the Password Authentication Factor, according to the AGD guidance. The evaluator shall change the BAF according to the AGD guidance and verify that the TSF requires entry of the Password Authentication Factor before allowing the factor to be changed.

Test 48: [conditional] If hybrid is selected in FIA_UAU.5.1, the evaluator shall configure the TSF to use the BAF and PIN or password, which includes configuring the Password Authentication Factor, according to the AGD guidance. The evaluator shall change the BAF and PIN according to the AGD guidance and verify that the TSF requires the entry of the Password Authentication Factor before allowing the factor to be changed.

Test 46 - See FIA_PMG_EXT.1 test 43 for the base device and for the Knox container which require the previous password before allowing the password to be changed.

Test 47 - The evaluator enrolled a new fingerprint and was required to enter the base device password to enroll fingerprints.

Test 48 – This is addressed by the previous two test cases where the same password and the same enrolled biometric are used for the hybrid authentication, except they must be used together. In the case of Knox, the evaluator tested that when changing any authentication factor the evaluator had to enter the current password (regardless of current authentication type) and then the password or hybrid authentication mechanisms, including ensuring configuration of the available biometric factors could be performed.

2.4.17 Re-Authenticating (TSF Lock) (MDFPP33:FIA_UAU.6/LOCKED)

2.4.17.1 MDFPP33:FIA_UAU.6.1/LOCKED

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Test 49: The evaluator shall configure the TSF to transition to the locked state after a time of inactivity (FMT_SMF.1) according to the AGD guidance. The evaluator shall wait until the TSF locks and then verify that the TSF requires the entry of the Password Authentication Factor before transitioning to the unlocked state.

Test 50: [conditional] If biometric in accordance with the Biometric Enrollment and Verification, version 1.1 is selected in FIA_UAU.5.1, for each BAF claimed in FIA_MBV_EXT.1.1 in the Biometric Enrollment and Verification, version 1.1 the evaluator shall repeat Test 1 verifying that the TSF requires the entry of the BAF before transitioning to the unlocked state.

Test 51: [conditional] If hybrid is selected in FIA_UAU.5.1, the evaluator shall repeat Test 1 verifying that the TSF requires the entry of the BAF and PIN/password before transitioning to the unlocked state.

Test 52: The evaluator shall configure user-initiated locking according to the AGD guidance. The evaluator shall lock the TSF and then verify that the TSF requires the entry of the Password Authentication Factor before transitioning to the unlocked state.

Test 53: [conditional] If biometric in accordance with the Biometric Enrollment and Verification, version 1.1 is selected in FIA_UAU.5.1, for each BAF claimed in FIA_MBV_EXT.1.1 in the Biometric Enrollment and Verification, version 1.1 the evaluator shall repeat Test 52 verifying that the TSF requires the entry of the BAF before transitioning to the unlocked state.

Test 54: [conditional] If hybrid is selected in FIA_UAU.5.1, the evaluator shall repeat Test 52 verifying that the TSF requires the entry of the BAF and PIN/password before transitioning to the unlocked state.

Test 1 - See FTA_SSL_EXT.1 test 108 where session timeouts are tested showing the device transitions to a locked state. This is the same regardless of authentication mechanism. See FIA_UAU.5 test 44 where all of the authentication rules are tested for the device, including re-authenticating when the device is locked.

KNOX - See FTA_SSL_EXT.1 test 108 where session timeouts are tested showing the Knox container transitions to a locked state. This is the same regardless of authentication mechanism. See FIA_UAU.5 test 44 where all of the authentication rules are tested for the Knox containers, including re-authenticating when the device is locked.

Test 2 - See FTA_SSL_EXT.1 test 108 where session timeouts are tested showing the device transitions to a locked state. This is the same regardless of authentication mechanism. See FIA_UAU.5 test 44 where all of the authentication rules are tested, including re-authenticating when the device is locked.

KNOX – Not applicable.

Test 3 – Not applicable.

KNOX: See FTA_SSL_EXT.1 test 108 where session timeouts are tested showing the device and Knox container transition to a locked state. This is the same regardless of authentication mechanism. See FIA_UAU.5 test 1 where all of the authentication rules are tested for the device and Knox containers, including re-authenticating when the Knox container is locked.

Test 4 - See FTA_SSL_EXT.1 test 109 where the device and Knox container are manually locked. This is the same regardless of authentication mechanism. See FIA_UAU.5 test 44 where all of the authentication rules are tested, including re-authenticating when the device is locked.

KNOX**:** See Test Case FTA_SSL_EXT.1 test 109 where the device and Knox container are manually locked. This is the same regardless of authentication mechanism. See FIA_UAU.5 test 44 where all of the authentication rules are tested, including re-authenticating when the Knox container is locked.

Test 5 - See FTA_SSL_EXT.1 test 109 where the device and Knox container are manually locked. This is the same regardless of authentication mechanism. See FIA_UAU.5 test 44 where all of the authentication rules are tested, including re-authenticating when the device is locked.

KNOX: Not applicable.

Test 6 - Not applicable.

KNOX: See FTA_SSL_EXT.1 test 109 where the device and Knox container are manually locked. This is the same regardless of authentication mechanism. See FIA_UAU.5 test 44 where all of the authentication rules are tested, including re-authenticating when the device is locked.

2.4.18 Protected Authentication Feedback (MDFPP33:FIA_UAU.7)

2.4.18.1 MDFPP33:FIA_UAU.7.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS describes the means of obscuring the authentication entry, for all authentication methods specified in FIA_UAU.5.1.

Section 6.4 of the ST explains the two lock screens (device lock screen and work profile lock screen), by default, briefly display the most recently entered password character and then obscures the character by replacing the displayed character with a dot symbol. The user can configure the TOE's behavior for the device lock screen so that it does not briefly display the last typed character; however, the TOE always briefly displays the last entered character for the work profile lock screen. Additionally, the TOE's device lock screen does not provide any feedback other than a notification of a failed biometric (fingerprint) authentication attempt ("not recognized"). Similarly, the TOE's work profile lock screen, when configured for hybrid authentication, displays only an indication ("no match") of a failed biometric attempt.

Work Profile – Section 6.9 of the ST explains work profile allows the user to enter the user's password from the lock screen. The work profile will, by default, display the most recently entered character of the password briefly or until the user enters the next character in the password, at which point the work profile obscures the character by replacing the character with a dot symbol.

The work profile can also be configured for hybrid authentication. In this case the user is prompted to first scan their biometric successfully before being prompted to enter their password. If the biometric attempt is unsuccessful, the user is prompted again to enter the biometric. To successfully login, both a valid biometric and the correct password must be entered.

Component Guidance Assurance Activities: The evaluator shall verify that any configuration of this requirement is addressed in the AGD guidance and that the password is obscured by default.

The password is obscured by default so no configuration is necessary.

Component Testing Assurance Activities: Test 55: The evaluator shall enter passwords on the device, including at least the Password Authentication Factor at lockscreen, and verify that the password is not displayed on the device.

Test 56: [conditional] If biometric in accordance with the Biometric Enrollment and Verification, version 1.1 is selected in FIA_UAU.5.1, for each BAF claimed in FIA_MBV_EXT.1.1 in the Biometric Enrollment and Verification, version 1.1 the evaluator shall authenticate by producing a biometric sample at lock screen. As the biometric algorithms are performed, the evaluator shall verify that sensitive images, audio, or other information identifying the user are kept secret and are not revealed to the user. Additionally, the evaluator shall produce a biometric sample that fails to authenticate and verify that the reasons for authentication failure (user mismatch, low sample quality, etc.) are not revealed to the user. It is acceptable for the BAF to state that it was unable to physically read the biometric sample, for example, if the sensor is unclean or the biometric sample was removed too quickly. However, specifics regarding why the presented biometric sample failed authentication shall not be revealed to the user.

Test 55 –The evaluator observed that at the start-up screen, device unlock screen, password confirmation screen, and change password screen no characters were displayed to the user.

KNOX - The evaluator entered a password at the KNOX login screen and observed that it was not echoed back when confirming the password prior to changing the password. The evaluator also observed while unlocking a KNOX container and changing a KNOX container password that passwords were not echoed when entering a container and when being changed.

Test 56 - This was tested as part of FIA_AFL_EXT.1 test 42 where fingerprints were tested for the base device and where hybrid was tested for Knox containers. The evaluator observed no visual or audio responses when entering fingerprints

2.4.19 Authentication for Cryptographic Operation (MDFPP33:FIA_UAU_EXT.1)

2.4.19.1 MDFPP33:FIA_UAU_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS section of the ST describes the process for decrypting protected data and keys. The evaluator shall ensure that this process requires the user to enter a Password Authentication Factor and, in accordance with FCS_CKM_EXT.3, derives a KEK, which is used to protect the software-based secure key storage and (optionally) DEK(s) for sensitive data, in accordance with FCS_STG_EXT.2.

Section 6.4 of the ST explains that the TOE's Key Hierarchy requires the user's password in order to derive the sHEK in order to decrypt other KEKs and DEKs. Thus, until it has the user's password, the TOE cannot decrypt the DEK utilized by FBE to decrypt protected data.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The following tests may be performed in conjunction with FDP_DAR_EXT.1 and FDP_DAR_EXT.2.

Evaluation Activity Note: The following test require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

Test 57: The evaluator shall enable encryption of protected data and require user authentication according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that includes a unique string treated as protected data.

The evaluator shall reboot the device, use a tool provided by developer to search for the unique string amongst the application data, and verify that the unique string cannot be found. The evaluator shall enter the Password Authentication Factor to access full device functionality, use a tool provided by the developer to access the unique string amongst the application data, and verify that the unique string can be found.

Test 58: [conditional] The evaluator shall require user authentication according to the AGD guidance. The evaluator shall store a key in the software-based secure key storage.

The evaluator shall lock the device, use a tool provided by developer to access the key amongst the stored data, and verify that the key cannot be retrieved or accessed. The evaluator shall enter the Password Authentication Factor to access full device functionality, use a tool provided by developer to access the key, and verify that the key can be retrieved or accessed.

Test 59: [conditional] The evaluator shall enable encryption of sensitive data and require user authentication according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that includes a unique string treated as sensitive data.

The evaluator shall lock the device, use a tool provided by developer to attempt to access the unique string amongst the application data, and verify that the unique string cannot be found. The evaluator shall enter the Password Authentication Factor to access full device functionality, use a tool provided by developer to access the unique string amongst the application data, and verify that the unique string can be retrieved.

Test 57 – The developer provided a test application that created known data in a device that has encrypted internal storage. The evaluator searched the internal storage and could not find the known value. The string can be found in applicable files only after the device has been unlocked with the user's password. The evaluator demonstrated the SD card can be read after the password is entered but cannot be read otherwise.

Test 58 –The evaluator imported a rootca and private Wi-Fi certificate into the device. The evaluator used an application to search for the known values and could not find the values on the internal storage. The evaluator then used an application to generate a known key stored in the keystore. The evaluator rebooted the device, demonstrated the key was not accessible before logging on, logged on, and demonstrated using an application that the known key could be found.

Test 59 - The devices including the S24 FE and Z Flip6 5G use the NIAPSEC library for sensitive data protection. For those devices, the evaluator created sensitive data through the vendor’s sensitive data protection API and attempted to access this file while the device was unlocked and locked. The evaluator found that while sensitive data can be written at any time, in order to read the sensitive data, the evaluator had to supply an authentication factor. With the correct authentication factor, the TOE was able to provide access to the protected keystore and the necessary keys could be used to decrypt the sensitive data. Without the password, the evaluator was not able to read the contents of the sensitive data.

The A53 and A52 devices use the Knox container to store sensitive data automatically. For those devices the evaluator used a procedure and adb shell commands to alternately attempt to access and decrypt sensitive data when the device is locked and unlocked. The evaluator verified that when the TOE is locked, the attempts fail and when it is unlocked, the attempts succeed.

2.4.20 Timing of Authentication (MDFPP33:FIA_UAU_EXT.2)

2.4.20.1 MDFPP33:FIA_UAU_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.20.2 MDFPP33:FIA_UAU_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the actions allowed by unauthorized users in the locked state.

The ST Section 6.4 states that the TOE, when configured to require a user password (as is the case in CC mode), allows a user to perform the actions described in section 5.1.4.21 of the ST. Beyond those actions, a user cannot perform any other actions other than observing notifications displayed on the lock screen until after successfully authenticating.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall attempt to perform some actions not listed in the selection while the device is in the locked state and verify that those actions do not succeed.

Test – The evaluator locked the device and attempted to perform some functions that could not be done without entering a password. These functions were Bluetooth, Wi-Fi, Airplane Mode, Secure Folder, Smart View, attempting to shut the device off, and finally attempting to enable emergency mode. As expected, none of these functions could be completed without the device being unlocked with a password.

2.4.21 Secondary User Authentication (MDFPP33:FIA_UAU_EXT.4)

2.4.21.1 MDFPP33:FIA_UAU_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.21.2 MDFPP33:FIA_UAU_EXT.4.2

TSS Assurance Activities: The evaluator shall verify that the TSS section of the ST describes the process for decrypting Enterprise application data and shared resource data. The evaluator shall ensure that this process requires the user to enter an Authentication Factor and, in accordance with FCS_CKM_EXT.3, derives a KEK which is used to protect the software-based secure key storage and (optionally) DEKs for sensitive data, in accordance with FCS_STG_EXT.2.

Section 6.9 of the ST explains the TOE requires a separate password or hybrid authentication for its work profile, thus protecting all Enterprise application data and shared resource data. The user must enter either their password or both their password and biometric (if the user has configured hybrid authentication and enrolled a biometric) in order to access any of the Enterprise application data. More detail is provided in the KMD for this processing.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The Evaluation Activities for any selected requirements related to device authentication must be separately performed for the secondary authentication mechanism (in addition to activities performed for the primary authentication mechanism). The requirements are: FIA_UAU.6, FIA_PMG_EXT.1, FIA_TRT_EXT.1, FIA_UAU.7, FIA_UAU_EXT.2, FTA_SSL_EXT.1, FCS_STG_EXT.2, FMT_SMF.1/FMT_MOF_EXT.1 #1, #2, #8, #21, and #36.

Additionally, FIA_AFL_EXT.1 must be met, except that in FIA_AFL_EXT.1.2 the separate test is performed with the text 'wipe of all protected data' changed to 'wipe of all Enterprise application data and all Enterprise shared resource data.'

All of the requirements have taken the secondary authentication mechanism into consideration.

FIA_UAU.6 – see FIA_UAU.6/CREDENTIAL test case 48

FIA_PMG_EXT.1 – see FIA_PMG_EXT.1 test case 43 (Knox)

FIA_TRT_EXT.1 - FIA_TRT_EXT.1 (not applicable, no test cases)

FIA_UAU.7 –FIA_UAU.7 test case 55 (Knox screens)

FIA_UAU_EXT.2 - FIA_UAU_EXT.2 test case 1 (not applicable, any non-container activities can be performed outside the container, the evaluator has found that the container contents cannot be access until unlocked via other tests performed)

FTA_SSL_EXT.1 – see FTA_SSL_EXT.1 test case 108 and FTA_SSL_EXT.1 test case 109 (Knox container variations)

FCS_STG_EXT.2 - FCS_STG_EXT.2 (not applicable, no test cases)

FMT_SMF_EXT.1/FMT_MOF_EXT.1 #1 – see FMT_MOF_EXT.1/Knox test case 70, function 1 & FMT_SMF_EXT.1/Knox function 1

FMT_SMF_EXT.1/FMT_MOF_EXT.1 #2 – see FMT_MOF_EXT.1/Knox- test case 70, function 2 & FMT_SMF_EXT.1/Knox function 2

FMT_SMF_EXT.1/FMT_MOF_EXT.1 #8 – see FMT_MOF_EXT.1/Knox test case 70, function 8 & FMT_SMF_EXT.1/Knox function 8

FMT_SMF_EXT.1/FMT_MOF_EXT.1 #21 - FMT_MOF_EXT.1- test case 70, function 21 & FMT_SMF_EXT.1-function 21 (not applicable, removable media is not accessible from the Knox container)

FMT_SMF_EXT.1/FMT_MOF_EXT.1 #36 - FMT_MOF_EXT.1- test case 70, function 36 & FMT_SMF_EXT.1-function 36 (not applicable, there is no separate banner for the enterprise container inside the device)

FIA_AFL.1 – see FIA_AFL_EXT.1/Knox-test case 41a, FIA_AFL_EXT.1/Knox-test case 41b, & FIA_AFL_EXT.1/Knox-test case 42

2.4.22 X.509 Validation of Certificates - per TD0689 (MDFPP33:FIA_X509_EXT.1)

2.4.22.1 MDFPP33:FIA_X509_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.22.2 MDFPP33:FIA_X509_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

Section 6.4 of the ST explains the TOE checks the validity of all imported CA certificates by checking for the presence of the basicConstraints extension and that the CA flag is set to TRUE as the TOE imports the certificate into the TOE's Trust Anchor Database. If the TOE detects the absence of either the extension or flag, the TOE will import the certificate as a user public key and add it to the keystore (not the Trust Anchor Database). The TOE also checks all the certificates in the server's certificate chain (including the server's certificate) for the presence of the basicConstraints extension and CA flag in each CA certificate. Similarly, the TOE verifies the extendedKeyUsage Server Authentication purpose during certificate validation. The TOE's certificate validation algorithm examines each certificate in the path (starting with the peer's certificate) and first checks for validity of that certificate (e.g., has the certificate expired? or is it not yet valid? whether the certificate contains the appropriate X.509 extensions [e.g., the CA flag in the basic constraints extension for a CA certificate, or that a server certificate contains the Server Authentication purpose in the extendedKeyUsage field]), then verifies each certificate in the chain (applying the same rules as above, but also ensuring that the Issuer of each certificate matches the Subject in the next rung "up" in the chain and that the chain ends in a self-signed certificate present in either the TOE's trusted anchor database or matches a specified Root CA), and finally the TOE performs revocation checking for all certificates in the chain.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The tests described must be performed in conjunction with the other Certificate Services evaluation activities, including the use cases in FIA_X509_EXT.2.1 and FIA_X509_EXT.3. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. The evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

Test 60: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn:

- By establishing a certificate path in which one of the issuing certificates is not a CA certificate,

- By omitting the basicConstraints field in one of the issuing certificates,

- By setting the basicConstraints field in an issuing certificate to have CA=False,

- By omitting the CA signing bit of the key usage field in an issuing certificate, and

- By setting the path length field of a valid CA field to a value strictly less than the certificate path.

The evaluator shall then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator shall then remove trust in one of the CA certificates, and show that the function fails.

Test 61: The evaluator shall demonstrate that validating an expired certificate results in the function failing.

Test 62: The evaluator shall test that the TOE can properly handle revoked certificates-conditional on whether CRL, OCSP, OSCP stapling, or OCSP multi-stapling is selected; if multiple methods are selected, then the following tests shall be performed for each method:

The evaluator shall test revocation of the node certificate.

The evaluator shall also test revocation of the intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA). For the test of the WLAN use case, only pre-stored CRLs are used. If OCSP stapling per RFC 6066 is the only supported revocation method, this test is omitted.

The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.

Test 63: If any OCSP option is selected, the evaluator shall configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL as specified in RFC 8603 is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set, and verify that validation of the CRL fails.

Test 64: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate (the certificate will fail to parse correctly).

Test 65: The evaluator shall modify any bit in the last byte of the signature algorithm of the certificate and demonstrate that the certificate fails to validate (the signature on the certificate will not validate).

Test 66: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate (the signature on the certificate will not validate).

Test 67:

Test 67.1: (Conditional on support for EC certificates as indicated in FCS_COP.1(3)). The evaluator shall establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

Test 67.2: (Conditional on support for EC certificates as indicated in FCS_COP.1(3)). The evaluator shall replace the intermediate certificate in the certificate chain for Test 8a with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 8a, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

All test cases were run in for the following protocols: HTTPS, TLS, IPSec.

Test 60 – The evaluator configured valid server and client certificates. A successful connection was made in each case. The evaluator then configured a server certificate with an invalid certification path by deleting the root CA so that the certificate chain was invalid because of a missing (or deleted) certificate. The connection was refused in each case. The evaluator then configured a test server to send an authentication certificate issued by a Sub CA with no BasicConstraints and with BasicConstraints but the CA Flag set to false and then with an intermediate CA with no keyCertSign purpose and then with an intermediate CA with a path length field set too low. The connection was refused in each case.

Test 61 – The evaluator configured valid server and client certificates. A successful connection was made in each case. The evaluator then configured a server certificate that was expired. The connection was refused in each case. The evaluator then configured a server certificate that had an expired subCA. The connection was refused in each case.

Test 62 – The evaluator configured valid server and client certificates. A successful connection was made in each case. The evaluator then configured a server certificate that was revoked and then a server certificate issued by an intermediate CA that is revoked. The connection was refused in each case for CRL (in the case of HTTPS, TLS) and OCSP (in the case of HTTPS, TLS and all IKE combinations).

Test 63 – The evaluator configured valid server and client certificates. A successful connection was made in each case. The evaluator then configured a server certificate issued by an intermediate CA referring to a CRL revocation server where the signer lacks cRLSign or OCSPSigning, and one issued by an intermediate CA whose issuer CA refers to a CRL revocation server where the signer lacks cRLSign or OCSPSigning. The connection was refused in each case for CRL (in the case of HTTPS, TLS) and OCSP (in the case of HTTPS, TLS and all IKE combinations).

Test 64- The evaluator configured valid server and client certificates. A successful connection was made in each case. The evaluator then configured the server to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. The connection was refused in each case.

Test 65 – This test was performed with test 64.

Test 66 – This test was performed with test 64.

Test 67.1 – The evaluator configured valid server and client certificates. A successful connection was made in each case. The evaluator then configured the server to send an authentication certificate with an explicitly defined elliptic curve. The connection was refused in each case.

Test 67.2 – This was tested with 67.1.

2.4.23 X.509 Certificate Validation (WLANC10:FIA_X509_EXT.1/WLAN)

2.4.23.1 WLANC10:FIA_X509_EXT.1.1/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.23.2 WLANC10:FIA_X509_EXT.1.2/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes where the check of validity of the EAP-TLS certificates takes place. The evaluator shall also ensure the TSS also provides a description of the certificate path validation algorithm.

Section 6.4 of the ST explains the TOE checks the validity of all imported CA certificates by checking for the presence of the basicConstraints extension and that the CA flag is set to TRUE as the TOE imports the certificate into the TOE's Trust Anchor Database. If the TOE detects the absence of either the extension or flag, the TOE will import the certificate as a user public key and add it to the keystore (not the Trust Anchor Database). The TOE also checks all the certificates in the server's certificate chain (including the server's certificate) for the presence of the basicConstraints extension and CA flag in each CA certificate. Similarly, the TOE verifies the extendedKeyUsage Server Authentication purpose during certificate validation. The TOE's certificate validation algorithm examines each certificate in the path (starting with the peer's certificate) and first checks for validity of that certificate (e.g., has the certificate expired? or is it not yet valid? whether the certificate contains the appropriate X.509 extensions [e.g., the CA flag in the basic constraints extension for a CA certificate, or that a server certificate contains the

Server Authentication purpose in the extendedKeyUsage field]), then verifies each certificate in the chain (applying the same rules as above, but also ensuring that the Issuer of each certificate matches the Subject in the next rung “up” in the chain and that the chain ends in a self-signed certificate present in either the TOE’s trusted anchor database or matches a specified Root CA), and finally the TOE performs revocation checking for all certificates in the chain.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The tests described must be performed in conjunction with the other Certificate Services assurance activities. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. The evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

Test 1: The evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function (e.g. application validation), and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.

Test 3: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.

Test 4: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension not set. The validation of the certificate path fails.

Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate (the certificate will fail to parse correctly).

Test 6: The evaluator shall modify any bit in the last byte of the signature algorithm of the certificate and demonstrate that the certificate fails to validate (the signature on the certificate will not validate).

Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate (the signature on the certificate will not validate).

Test 1 – The evaluator configured valid server and client certificates. A successful connection was made. The evaluator then configured a server certificate with an invalid certification path by deleting the root CA so that the certificate chain was invalid because of a missing (or deleted) certificate. The connection was refused.

Test 2 – The evaluator configured valid server and client certificates. A successful connection was made. The evaluator then configured a server certificate that was expired. The connection was refused. The evaluator then configured a server certificate that had an expired subCA. The connection was refused.

Test 3 – The evaluator configured a test server to send an authentication certificate issued by a root CA with no BasicConstraints. The connection was refused. The evaluator then configured a test server to send an authentication certificate issued by a root CA with BasicConstraints flag set to false. The connection was refused. The evaluator then configured a test server to send an authentication certificate with an intermediate CA with no keyCertSign purpose. The connection was refused.   The evaluator then configured a test server to send an authentication certificate with a bad path length. The connection was refused.

Test 4 – This test has been performed in test 3.

Test 5- The evaluator configured valid server and client certificates. A successful connection was made in each case. The evaluator then configured the server to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. The connection was refused in each case.

Test 6 – This test was performed with test 5.

Test 7 – This test was performed with test 5.

2.4.24 X.509 Certificate Authentication (MDFPP33:FIA_X509_EXT.2)

2.4.24.1 MDFPP33:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.24.2 MDFPP33:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described.

Section 6.4 of the ST explains the TOE uses X.509v3 certificates as part of EAP-TLS, TLS, HTTPS and IPsec authentication. The TOE comes with a built-in set of default Trusted Credentials (Android's set of trusted CA certificates). While the user cannot remove any of the built-in default CA certificates, the user can disable any of those certificates through the user interface so that certificates issued by disabled CAs cannot validate successfully. In addition, a user can import a new trusted CA certificate into the Keystore or an administrator can install a new certificate through an MDM.

The TOE does not establish TLS or HTTPS connections itself (beyond EAP-TLS used for WPA3/WPA2 Wi-Fi connections), but provides a series of APIs that mobile applications can use to check the validity of a peer certificate. When establishing an EAP-TLS connection, the TOE does not check for certificate revocation as the revocation servers are not available until after the EAP-TLS connection is established. The mobile application, after correctly using the specified APIs, can be assured as to the validity of the peer certificate and will not establish the trusted connection if the peer certificate cannot be verified (including validity, certification path, and revocation [through CRL and OCSP]).

The VPN requires that for each VPN profile, the user specify the client certificate the TOE will use (the certificate must have been previously imported into the keystore) and specify the CA certificate to which the server's certificate must chain. The VPN thus uses the specified certificate when attempting to establish that VPN connection. When establishing a connection to a VPN server, the VPN first compares the Identification (ID) Payload received from the server against the certificate sent by the server, and if the DN of the certificate does not match the ID, then the TOE does not establish the connection.

The TOE supports both CRL and OCSP configurations for revocation checking. Only one of these will be used at a time (even if both are configured). OCSP takes precedence over CRL if a certificate provides information for checking both. The process of checking the revocation status of a certificate depends on the configuration on the device set by the admin.

If OCSP is configured (and if the Authority Information Access, AIA, extension is present), OCSP will be used to check the status. If the certificate lacks AIA, or if OCSP is not configured on the device, the TOE attempts to determine revocation status using CRLs, if the certificate includes a CRL Distribution Point (CDP). If the TOE cannot establish a connection with the server acting as the CDP or OCSP server, or does not receive a positive confirmation from the OCSP server, the TOE will deem the server's certificate as invalid and not establish a TLS connection with the server. Note that the VPN only checks OCSP for revocation (if it is configured on the device).

Component Guidance Assurance Activities: If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

The requirement is not for the administrator to be able to specify the default action so no guidance is required.

Component Testing Assurance Activities: The evaluator shall perform the following test for each trusted channel:

Test 68: The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the operational guidance to determine that all supported administrator-configurable options behave in their documented manner.

Test 68 – The evaluator made a valid HTTPS connection. The evaluator then attempted to access an unavailable revocation server and the connection was rejected as expected. Testing was performed for both HTTPS and TLS as well as both OCSP and CRLs. The evaluator repeated the test for two IPsec variations (IKEv2/RSA, IKEv2/ECDSA). The evaluator made a valid IPsec connection. The evaluator then defined a non-existent revocation server and attempted the connection a second time and the connection was rejected as expected. Only OSCP is supported for IPsec.

2.4.25 X.509 Certificate Authentication (VPNC24:FIA_X509_EXT.2)

2.4.25.1 VPNC24:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.25.2 VPNC24:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described.

See MDFPP33:FIA_X509_EXT.2

Component Guidance Assurance Activities: If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

See MDFPP33:FIA_X509_EXT.2

Component Testing Assurance Activities: Refer to the EAs for FIA_X509_EXT.2 in the MDF PP (included below). The only change to this SFR is that some selections are mandated, therefore the corresponding testing is mandatory. The actual testing for those selections is not changed.

The evaluator shall perform the following test for each trusted channel:

Test 1: The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the operational guidance to determine that all supported administrator-configurable options behave in their documented manner.

See MDFPP33:FIA_X509_EXT.2

2.4.26 X.509 Certificate Authentication (EAP-TLS for WLAN) - TD0703 applied (WLANC10:FIA_X509_EXT.2/WLAN)

2.4.26.1 WLANC10:FIA_X509_EXT.2.1/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operational environment so that the TOE can use the certificates. (TD0703 applied)

See MDFPP33:FIA_X509_EXT.2

Component Guidance Assurance Activities: If not already present in the TSS, the evaluator shall check the administrative guidance to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions for configuring the operating environment so that the TOE can use the certificates. (TD0703 applied)

See MDFPP33:FIA_X509_EXT.2

Component Testing Assurance Activities: Removed per TD0703.

2.4.27 Request Validation of Certificates (MDFPP33:FIA_X509_EXT.3)

2.4.27.1 MDFPP33:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.27.2 MDFPP33:FIA_X509_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall verify that the API documentation provided according to Section 5.2.2 Class ADV: Development includes the security function (certificate validation) described in this requirement. This documentation shall be clear as to which results indicate success and failure.

Section 6.1 of the Admin Guide provides APIs for certificate verification checking.

Component Testing Assurance Activities: The evaluator shall write, or the developer shall provide access to, an application that requests certificate validation by the TSF. The evaluator shall verify that the results from the validation match the expected results according to the API documentation. This application may be used to verify that import, removal, modification, and validation are performed correctly according to the tests required by FDP_STG_EXT.1, FTP_ITC_EXT.1, FMT_SMF.1, and FIA_X509_EXT.1.

The developer provided an application that checked the following:

- Valid certificate paths can be successfully validated
- Invalid certificate paths are rejected
  - Certificate has no CRL distribution point and no OCSP responder location
  - Intermediate certificate is expired
  - Intermediate certificate is not yet valid
  - Both CRL server and OCSP responder are unavailable
  - Intermediate certificate is revoked by CRL (OCSP is unavailable)
  - Leaf certificate is revoked by CRL (OCSP is unavailable)
  - Intermediate certificate is revoked by CRL (OCSP responder location is absent)
  - Leaf certificate has damaged byte in the first eight bytes
  - Leaf certificate is revoked by CRL (OCSP responder location is absent)
  - Intermediate certificate does not have basicConstraints extension
  - Intermediate certificate has cA flag in basicConstraints extension set to FALSE
  - Leaf certificate has damaged public key byte
  - Leaf certificate is not signed by Intermediate certificate
  - Intermediate certificate is not signed by Root certificate
  - Root certificate is absent in Trust Anchor Database
  - Leaf certificate has damaged bit in the last byte of the signature algorithm
  - Intermediate certificate is revoked by OCSP
  - Leaf certificate is revoked by OCSP
  - OCSP Response for leaf certificate is signed by invalid authorized responder's certificate without OCSPSign OID (CRL not specified)
  - CRL for leaf certificate is signed with invalid certificate without cRLSign bit set (OCSP not specified)
  - OCSP Response for leaf certificate is signed by valid authorized responder's certificate (CRL not specified)
  - CRL for leaf certificate is signed with valid certificate (OCSP not specified)

2.4.28 Certificate Storage and Management (WLANC10:FIA_X509_EXT.6)

2.4.28.1 WLANC10:FIA_X509_EXT.6.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.28.2 WLANC10:FIA_X509_EXT.6.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it describes all certificate stores implemented that contain certificates used to meet the requirements of this PP-Module. This description shall contain information pertaining to how certificates are loaded into the store, and how the store is protected from unauthorized access.

If the TOE relies on a platform mechanism for certificate loading and storage, the evaluator shall verify that the TSS identifies this mechanism and describes how use of this mechanism is protected against unauthorized access.

Section 6.4 of the ST explains the TOE uses X.509v3 certificates as part of EAP-TLS, TLS, HTTPS and IPsec authentication. The TOE comes with a built-in set of default Trusted Credentials (Android's set of trusted CA certificates). While the user cannot remove any of the built-in default CA certificates, the user can disable any of those certificates through the user interface so that certificates issued by disabled CAs cannot validate successfully. In addition, a user can import a new trusted CA certificate into the Keystore or an administrator can install a new certificate through an MDM.

The TOE does not establish TLS or HTTPS connections itself (beyond EAP-TLS used for WPA3/WPA2 Wi-Fi connections), but provides a series of APIs that mobile applications can use to check the validity of a peer certificate. When establishing an EAP-TLS connection, the TOE does not check for certificate revocation as the revocation servers are not available until after the EAP-TLS connection is established. The mobile application, after correctly using the specified APIs, can be assured as to the validity of the peer certificate and will not establish the trusted connection if the peer certificate cannot be verified (including validity, certification path, and revocation [through CRL and OCSP]).

The VPN requires that for each VPN profile, the user specify the client certificate the TOE will use (the certificate must have been previously imported into the keystore) and specify the CA certificate to which the server's certificate must chain. The VPN thus uses the specified certificate when attempting to establish that VPN connection. When establishing a connection to a VPN server, the VPN first compares the Identification (ID) Payload received from the server against the certificate sent by the server, and if the DN of the certificate does not match the ID, then the TOE does not establish the connection.

Component Guidance Assurance Activities: The evaluator shall check the administrative guidance to ensure that it describes how to load X.509 certificates into the TOE's certificate store, regardless of whether the TSF provides this mechanism itself or the TOE relies on a platform-provided mechanism for this.

Section 3.3 of the Admin Guide provides a spreadsheet with includes the MDM setting for how to import certificates.

Component Testing Assurance Activities: The evaluator shall perform the following test for each TOE function that requires the use of certificates:

Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. The evaluator shall then load any certificates needed to validate the certificate to be used in the function and demonstrate that the function succeeds. The evaluator shall then delete one of these dependent certificates and show that the function fails.

Test 2: The evaluator shall demonstrate that the mechanism used to load or configure X.509 certificates cannot be accessed without appropriate authorization.

Test 1 - See Test Case WLANCEP10:FIA_X509_EXT.1/WLAN test case 1.

Test 2 - See Test Case FIA_UAU_EXT.2 test case 1 where the evaluator showed that no other functions were allowed on the TOE while it was in the locked state (prior to authentication). See also Test Case FMT_SMF_EXT.1 test case 11 where the evaluator demonstrated the mechanism used to load X509 certificates.

2.5 Security management (FMT)

2.5.1 Management of Security Functions Behavior (MDFPP33:FMT_MOF_EXT.1)

2.5.1.1 MDFPP33:FMT_MOF_EXT.1.1

TSS Assurance Activities: The evaluator shall verify that the TSS describes those management functions that may only be performed by the user and confirm that the TSS does not include an Administrator API for any of these management functions. This activity will be performed in conjunction with FMT_SMF.1.

Sec 6.5 of the TSS references Table 8 within the ST which contains a table of all the management functions. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.2 MDFPP33:FMT_MOF_EXT.1.2

TSS Assurance Activities: The evaluator shall verify that the TSS describes those management functions that may be performed by the Administrator, to include how the user is prevented from accessing, performing, or relaxing the function (if applicable), and how applications/APIs are prevented from modifying the Administrator configuration. The TSS also describes any functionality that is affected by administrator-configured policy and how. This activity will be performed in conjunction with FMT_SMF.1.

The ST, Section 5.1.5.2, contains a table of all the management functions. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Test 69: The evaluator shall use the test environment to deploy policies to Mobile Devices.

Test 70: The evaluator shall create policies which collectively include all management functions which are controlled by the (enterprise) administrator and cannot be overridden/relaxed by the user as defined in FMT_MOF_EXT.1.2. The evaluator shall apply these policies to devices, attempt to override/relax each setting both as the user (if a setting is available) and as an application (if an API is available), and ensure that the TSF does not permit it. Note that the user may still apply a more restrictive policy than that of the administrator.

Test 71: Additional testing of functions provided to the administrator are performed in conjunction with the testing activities for FMT_SMF.1.1.

Test 69 – The developer provided an MDM application that was copied into local device storage and installed. The application is basically an MDM agent that allows the MDM functions to be performed through the UI. Once installed and enabled, the TOE device is effectively enrolled and policies can be defined and applied to the device.

Test 70 – The evaluator used an MDM application to set the following management restrictions to demonstrate management settings are enforced and the user could not override the settings. The test cases refer to the functions in the ST.

Function 1 – See the test case for FIA_PMG_EXT.1 where the evaluator tested with the password length configured to 16 characters (Minimum Length) and where the evaluator tested with the password length configured to 15 characters. The evaluator then configured an invalid numeric password sequence to test password complexity. The evaluator configured password expiration and reset the day to ensure it worked.

Knox – This test was repeated for Knox to ensure the passwords restrictions were enforced on containers.

Function 2 – The TOE does not allow screen lock to be disabled; only the timeout can be configured. The evaluator then restricted the maximum screen timeout to 3 minutes and verified that the screen lock and display timeout limits were restricted to no more than 3 minutes. Test Case FIA_AFL_EXT.1 test case 1 and test case 2 address testing of authentication failure limits. There are no user interfaces to set or change the authentication failure limits.

Knox – This test was repeated for Knox and session values were enforced as described in the test case.

Function 3a – The evaluator enabled and disabled the VPN to show the function was available or not as configured.

Function 3b – The evaluator configured a per-app policy for the VPN. The evaluator then attempted to use the app and found it could use the private VPN when the per-app policy was enabled.

Function 3c – This is the same as b, except that "Add All" can be selected or all applications can be individually selected. The previous test selected two apps each inside and outside the container, while configuring the VPN for the entire inside or outside of the container space simply entails selecting all possible applications in and/or out of the container. The set of applications inside or outside a container is the only applicable notion of groups of applications.

Knox – Knox is tested as part of Test case 3c.

Function 4 – The evaluator enabled and disabled each radio on a one-by-one basis. The evaluator observed functions with radios enabled and then observed (with the same configuration) absence of those functions when the associated radio is disabled. The evaluator enabled each radio, disallowed it as an administrator and ensured it could not be re-enabled by the user.

Function 5 – The evaluator ensured that the camera was usable. The evaluator then disabled the camera and ensured the camera was not usable. The evaluator repeated this with the microphone.

Knox – The test was repeated within the container with the same results.

Function 6 – No management restriction claimed.

Function 7 – No management restriction claimed.

Function 8a – The evaluator confirmed the Playstore was usable to installed applications. The evaluator then disabled the Playstore "Disable Market" and found that the Playstore application was removed and the evaluator even could not install Playstore applications via the website.

Knox – The device test ensures containers cannot install affected applications since the Playstore is not directly accessible from within a container.

Function 8b – The evaluator created an application blacklist. The evaluator demonstrated an application from the blacklist could not be installed while one not on the blacklist could. The evaluator then created an application whitelist. The evaluator demonstrated an application from the whitelist could be installed while one not on the whitelist could not.

Knox – The test was repeated within the container with the same results.

Function 8c – The evaluator changed the policy to disallow installation and uninstalled an application. The evaluator then attempted to reinstall the same application used for the whitelist test. The installation was denied.

Knox – The test was repeated within the container with the same results.

Function 9 – No management restriction claimed.

Function 10 – No management restriction claimed.

Function 11 – No management restriction claimed.

Function 12 – No management restriction claimed.

Function 13 – No management restriction claimed.

Function 14 – No management restriction claimed.

Function 15 – No management restriction claimed.

Function 16 – No management restriction claimed.

Function 17 – No management restriction claimed.

Function 18 – No management restriction claimed.

Function 19 – No management restriction claimed.

Function 20 – The evaluator encrypted the SD card as part of putting it into CC mode. The evaluator then reformatted and inserted an SD card, but without encrypting it, tried to access it. The evaluator found that the SD card could not be used until the SD card was encrypted. The evaluator proceeded to encrypt the SD card as required and found it could not be decrypted afterwards and its contents were accessible.

Function 21 – The evaluator ensured location services were on and could be used. The evaluator then turned off location services to see they could not be used – the application prompted the user to turn the services back on so they could be used. The evaluator turned the location services back on and then disabled them as the administrator. The evaluator then observed that location services were disabled and could not be turned on.

Function 22 – The evaluator unlocked the device with a fingerprint. The evaluator then disabled the fingerprint option for unlock and ensured that fingerprints could no longer be used to unlock the device. After unlocking the device with the password, the evaluator re-enabled the use of fingerprints through the Settings UI (as a user). The evaluator then disabled the use of fingerprints by using the MDM application as the administrator. The evaluator found that the user settings for fingerprints was disabled and the user was no longer able to unlock the TOE with fingerprints.

Knox – The test was repeated on the Knox container login using password and hybrid login settings. The evaluator ensured the container required the correct type of login.

Function 23 – No management restriction claimed.

Function 24 – No management restriction claimed.

Function 25 – No management restriction claimed.

Function 26 – Developer mode can be enabled from the device UI. The evaluator enabled developer mode and then enabled USB debugging, at which point the evaluator could use adb and was able to access and retrieve the screen shots on the device. The evaluator disabled USB debugging and was not able to issue adb commands to the device. The evaluator repeated the test using the MDM application. The evaluator disabled USB debugging using the MDM and verified the adb commands could not be used. The evaluator then re-enabled USB debugging and the adb commands worked as expected. The reboot portion of this test was performed in FMT_SMF_EXT.1 test 26.

Function 27 – For the purposes of the test, the evaluator configured the TOE to set the current location as a Trusted location so that when the TOE is in range of the trusted location, the local user authentication is not needed to unlock the device. Outside of the range, the user must perform authentication. The evaluator used the MDM to restrict this function. With the administrator restriction in place, the evaluator could not use authentication bypass service.

Function 28 – No management restriction claimed.

Function 29 – No management restriction claimed.

Function 30 – No management restriction claimed.

Function 31 – No management restriction claimed.

Function 32 – No management restriction claimed.

Function 33 – No management restriction claimed

Function 34 – No management restriction claimed.

Function 35 – No management restriction claimed.

Function 36 – This was tested as part of FTA_TAB.1. There is no user interface to configure the banner.

Function 37 – No management restriction claimed.

Function 38 – No management restriction claimed.

Function 39 – The evaluator first confirmed that MTP access was possible from an attached Windows computer. The evaluator then disabled MTP access and ensured that MTP was no longer usable. The evaluator then rebooted each TOE device into recovery mode and found the devices were not accessible.

Function 40 – No management restriction claimed.

Function 41a – The evaluator turned on each type of tethering and ensured that a connection could be made (Wi-Fi, USB, or a Bluetooth pairing). The evaluator then disabled each type of tethering using the MDM and demonstrated it was no longer available to the user. The evaluator enabled and configured a Wi-Fi hotspot with a pre-shared key. The evaluator attempted to connect with an incorrect pre-shared key and found that no connection could be made. The evaluator attempted to connect with the correct pre-shared key and found that the connection was established and usable. The TOE supports three types of Tethering: Wi-Fi, Bluetooth, and USB.

Function 41b – For USB tethering, the initial setup requires the user to actively approve the connection via authentication to the device.

Function 42 – No management restriction claimed.

> Knox - The test of this function is performed in conjunction with FDP_ACF_EXT.1.2 where the evaluator has tested that the administrator can restrict the ability to move data between the base device and Knox container.

Function 43 – This was tested as part of FMT_SMF_EXT.1, Function 43 where the evaluator created an app separation policy as an administrator and then demonstrated that installed applications were separated in accordance with the policy.

Function 44 – This was tested as part of FMT_SMF_EXT.2.1.

Function 45 – The evaluator configured a VPN connection and ensured it could be used to connect to a VPN gateway. The evaluator then disconnected the VPN and reconfigured it for Always-on to ensure that the TOE would automatically connect and stay connected. The evaluator then disabled the Always-On VPN feature as the administrator. At this point the evaluator found the TOE grayed out the Always-On VPN switch so it could not be attempted. The evaluator then re-enabled always on VPN and found it could once again be configured by the user.

Function 46 – No management restriction claimed.

Function 47a – The evaluator plugged in a USB Storage Host device (thumb drive) into the USB port on the TOE to ensure the function worked by examining the contents of the attached thumb drive. The evaluator then disabled USB Host Storage via policy (there are no user interfaces) and ensured that the attached USB thumb drive was no longer accessible (not visible). Attempts to reconnect a USB drive while blocked by policy yielded a security message.

Function 47b – The evaluator first ensured the TOE device was in CC mode. The evaluator then disabled CC mode. The evaluator then re-enabled CC mode.

Function 47c – The evaluator used the TOE's Date and Time settings to set the date on the device forward one day. Once updated, the evaluator reset the date. The evaluator then used the MDM App to disallow the user from changing the date and time settings. With this policy in place, the evaluator found that the date and time on the TOE device could not be changed as the option was greyed out.

Function 47d – The evaluator installed two applications and started both applications to show that they were able to be started. The evaluator then used the MDM app settings to disable the application so that it would no longer be able to start. The evaluator found that the application disappeared from the device and could not be started by the user. Since the installation of the application was not blocked, the evaluator tried to install the application again and start the application, but found that the security policy prevented the newly installed version of the app from running as well.

Test 71 - See test 2 above and its sub-tests and FMT_SMF_EXT.1.1 where administrator functions are tested.

2.5.2 Specification of Management Functions (MDFPP33:FMT_SMF.1)

2.5.2.1 MDFPP33:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes all management functions, what role(s) can perform each function, and how these functions are (or can be) restricted to the roles identified by FMT_MOF_EXT.1.

The following activities are organized according to the function number in the table. These activities include TSS Evaluation Activities, AGD Evaluation Activities, and test activities.

Test activities specified below shall take place in the test environment described in the evaluation activity for FPT_TUD_EXT.1.

Section 6.5 of the TSS references Table 8 within the ST which contains a table of all the management functions. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted

Component Guidance Assurance Activities: The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Section 3, Common Criteria Configuration, provides instructions for each of the functions claimed in FMT_SMF_EXT.1. The evaluator performed testing as both user and administrator as required. See the test AAs.

Component Testing Assurance Activities: Refer to the EAs for FMT_SMF_EXT.1 in the MDF PP (included below). The only change to this SFR is the change to management function 45. Testing of all other functions is not affected.

Function 1

The evaluator shall verify the TSS defines the allowable policy options: the range of values for both password length and lifetime, and a description of complexity to include character set and complexity policies (e.g., configuration and enforcement of number of uppercase, lowercase, and special characters per password).

Test 1: The evaluator shall exercise the TSF configuration as the administrator and perform positive and negative tests, with at least two values set for each variable setting, for each of the following:

- minimum password length

- minimum password complexity

- maximum password lifetime

Function 2

The evaluator shall verify the TSS defines the range of values for both timeout period and number of authentication failures for all supported authentication mechanisms.

Test 2: The evaluator shall exercise the TSF configuration as the administrator. The evaluator shall perform positive and negative tests, with at least two values set for each variable setting, for each of the following:

- screen-lock enabled/disabled

- screen lock timeout

- number of authentication failures (may be combined with test for FIA_AFL_EXT.1)

Function 3

Test 3: The evaluator shall perform the following tests:

a. The evaluator shall exercise the TSF configuration to enable the VPN protection. These configuration actions must be used for the testing of the FDP_IFC_EXT.1.1 requirement.

b. [conditional] If 'per-app basis' is selected, the evaluator shall create two applications and enable one to use the VPN and the other to not use the VPN. The evaluator shall exercise each application (attempting to access network resources; for example, by browsing different websites) individually while capturing packets from the TOE. The evaluator shall verify from the packet capture that the traffic from the VPN-enabled application is encapsulated in IPsec and that the traffic from the VPN-disabled application is not encapsulated in IPsec.

c. [conditional] If 'per-groups of application basis' is selected, the evaluator shall create two applications and the applications shall be placed into different groups. Enable one application group to use the VPN and the other to not use the VPN. The evaluator shall exercise each application (attempting to access network resources; for example, by browsing different websites) individually while capturing packets from the TOE. The evaluator shall verify from the packet capture that the traffic from the application in the VPN-enabled group is encapsulated in IPsec and that the traffic from the application in the VPN-disabled group is not encapsulated in IPsec.

Function 4

The evaluator shall verify that the TSS includes a description of each radio and an indication of if the radio can be enabled/disabled along with what role can do so. In addition the evaluator shall verify that the frequency ranges at which each radio operates is included in the TSS. The evaluator shall verify that the TSS includes at what point in the boot sequence the radios are powered on and indicates if the radios are used as part of the initialization of the device. The evaluator shall confirm that the AGD guidance describes how to perform the enable/disable function for each radio.

The evaluator shall ensure that minimal signal leakage enters the RF shielded enclosure (i.e. Faraday bag, Faraday box, RF shielded room) by performing the following steps:

Step 1: Place the antenna of the spectrum analyzer inside the RF shielded enclosure.

Step 2: Enable 'Max Hold' on the spectrum analyzer and perform a spectrum sweep of the frequency range between 300MHz â€“ 6000MHz, in I KHz steps (this range should encompass 802.11, 802.15, GSM, UMTS, and LTE). This range will not address NFC 13.56MHz, another test should be set up with similar constraints to address NFC.

If power above -90 dBm is observed, the Faraday box has too great of signal leakage and shall not be used to complete the test for Function 4.

Test 4: The evaluator shall exercise the TSF configuration as the administrator and, if not restricted to the administrator, the user, to enable and disable the state of each radio (e.g. Wi-Fi, cellular, NFC, Bluetooth). Additionally, the evaluator shall repeat the steps below, booting into any auxiliary boot mode supported by the device. For each radio, the evaluator shall:

Step 1: Place the antenna of the spectrum analyzer inside the RF shielded enclosure. Configure the spectrum analyzer to sweep desired frequency range for the radio to be tested (based on range provided in the TSS)). The ambient noise floor shall be set to -110dBm. Place the TOE into the RF shielded enclosure to isolate them from all other RF traffic.

Step 2: The evaluator shall create a baseline of the expected behavior of RF signals. The evaluator shall power on the device, ensure the radio in question is enabled, power off the device, enable 'Max Hold' on the spectrum analyzer and power on the device. The evaluator shall wait 2 minutes at each Authentication Factor interface prior to entering the necessary password to complete the boot process, waiting 5 minutes after the device is fully booted. The evaluator shall observe that RF spikes are present at the expected uplink channel frequency. The evaluator shall clear the 'Max Hold' on the spectrum analyzer.

Step 3: The evaluator shall verify the absence of RF activity for the uplink channel when the radio in question is disabled. The evaluator shall complete the following test five times. The evaluator shall power on the device, ensure the radio in question is disabled, power off the device, enable 'Max Hold' on the spectrum analyzer and power on the device. The evaluator shall wait 2 minutes at each Authentication Factor interface prior to entering the necessary password to complete the boot process, waiting 5 minutes after the device is fully booted. The evaluator shall clear the 'Max Hold' on the spectrum analyzer. If the radios are used for device initialization, then a spike of RF activity for the uplink channel can be observed initially at device boot. However, if a spike of RF activity for the uplink channel of the specific radio frequency band is observed after the device is fully booted or at an Authentication Factor interface it is deemed that the radio is enabled.

Function 5

The evaluator shall verify that the TSS includes a description of each collection device and an indication of if it can be enabled/disabled along with what role can do so. The evaluator shall confirm that the AGD guidance describes how to perform the enable/disable function.

Test 5: The evaluator shall perform the following test(s):

a. The evaluator shall exercise the TSF configuration as the administrator and, if not restricted to the administrator, the user, to enable and disable the state of each audio or visual collection devices (e.g. camera, microphone) listed by the ST author. For each collection device, the evaluator shall disable the device and then attempt to use its functionality. The evaluator shall reboot the TOE and verify that disabled collection devices may not be used during or early in the boot process. Additionally, the evaluator shall boot the device into each available auxiliary boot mode and verify that the collection device cannot be used.

b. [conditional] If 'per-app basis' is selected, the evaluator shall create two applications and enable one to use access the A/V device and the other to not access the A/V device. The evaluator shall exercise each application attempting to access the A/V device individually. The evaluator shall verify that the enabled application is able to access the A/V device and the disabled application is not able to access the A/V device.

c. [conditional] If 'per-groups of application basis' is selected, the evaluator shall create two applications and the applications shall be placed into different groups. Enable one group to access the A/V device and the other to not access the A/V device. The evaluator shall exercise each application attempting to access the A/V device individually. The evaluator shall verify that the application in the enabled group is able to access the A/V device and the application in the disabled group is not able to access the A/V device.

Function 6

Test 6: The evaluator shall use the test environment to instruct the TSF, both as a user and as the administrator, to command the device to transition to a locked state, and verify that the device transitions to the locked state upon command.

Function 7

Test 7: The evaluator shall use the test environment to instruct the TSF, both as a user and as the administrator, to command the device to perform a wipe of protected data. The evaluator must ensure that this management setup is used when conducting the Evaluation Activities in FCS_CKM_EXT.5.

Function 8

The evaluator shall verify the TSS describes the allowable application installation policy options based on the selection included in the ST. If the application allowlist is selected, the evaluator shall verify that the TSS includes a description of each application characteristic upon which the allowlist may be based.

Test 8: The evaluator shall exercise the TSF configuration as the administrator to restrict particular applications, sources of applications, or application installation according to the AGD guidance. The evaluator shall attempt to install unauthorized applications and ensure that this is not possible. The evaluator shall, in conjunction, perform the following specific tests:

a. [conditional] The evaluator shall attempt to connect to an unauthorized repository in order to install applications.

b. [conditional] The evaluator shall attempt to install two applications (one allowlisted, and one not) from a known allowed repository and verify that the application not on the allowlist is rejected. The evaluator shall also attempt to side-load executables or installation packages via USB connections to determine that the white list is still adhered to.

Function 9 & Function 10

The evaluator shall verify that the TSS describes each category of keys/secrets that can be imported into the TSF's secure key storage.

Test 9: The test of these functions is performed in association with FCS_STG_EXT.1.

Test 10: The test of these functions is performed in association with FCS_STG_EXT.1.

Function 11

The evaluator shall review the AGD guidance to determine that it describes the steps needed to import, modify, or remove certificates in the Trust Anchor database, and that the users that have authority to import those certificates (e.g., only administrator, or both administrators and users) are identified.

Test 11: The evaluator shall import certificates according to the AGD guidance as the user and/or as the administrator, as determined by the administrative guidance. The evaluator shall verify that no errors occur during import. The evaluator should perform an action requiring use of the X.509v3 certificate to provide assurance that installation was completed properly.

Function 12

The evaluator shall verify that the TSS describes each additional category of X.509 certificates and their use within the TSF.

Test 12: The evaluator shall remove an administrator-imported certificate and any other categories of certificates included in the assignment of function 14 from the Trust Anchor Database according to the AGD guidance as the user and as the administrator.

Function 13

The evaluator shall examine the TSS to ensure that it contains a description of each management function that will be enforced by the enterprise once the device is enrolled. The evaluator shall examine the AGD guidance to determine that this same information is present.

Test 13: The evaluator shall verify that user approval is required to enroll the device into management.

Function 14

The evaluator shall verify that the TSS includes an indication of what applications (e.g., user-installed applications, Administrator-installed applications, or Enterprise applications) can be removed along with what role can do so. The evaluator shall examine the AGD guidance to determine that it details, for each type of application that can be removed, the procedures necessary to remove those applications and their associated data. For the purposes of this Evaluation Activity, 'associated data' refers to data that are created by the app during its operation that do not exist independent of the app's existence, for instance, configuration data, or e-mail information that's part of an email client. It does not, on the other hand, refer to data such as word processing documents (for a word processing app) or photos (for a photo or camera app).

Test 14: The evaluator shall attempt to remove applications according to the AGD guidance and verify that the TOE no longer permits users to access those applications or their associated data.

Function 15

Test 15: The evaluator shall attempt to update the TSF system software following the procedures in the AGD guidance and verify that updates correctly install and that the version numbers of the system software increase.

Function 16

Test 16: The evaluator shall attempt to install an application following the procedures in the AGD guidance and verify that the application is installed and available on the TOE.

Function 17

Test 17: The evaluator shall attempt to remove any Enterprise applications from the device by following the administrator guidance. The evaluator shall verify that the TOE no longer permits users to access those applications or their associated data.

Function 18

The evaluator shall examine the AGD Guidance to determine that it specifies, for at least each category of information selected for Function 18, how to enable and disable display information for that type of information in the locked state.

Test 18: For each category of information listed in the AGD guidance, the evaluator shall verify that when that TSF is configured to limit the information according to the AGD, the information is no longer displayed in the locked state.

Function 19

Test 19: The evaluator shall exercise the TSF configuration as the administrator and, if not restricted to the administrator, the user, to enable system-wide data-at-rest protection according to the AGD guidance. The evaluator shall ensure that all Evaluation Activities for DAR (FDP_DAR) are conducted with the device in this configuration.

Function 20

Test 20: The evaluator shall exercise the TSF configuration as the administrator and, if not restricted to the administrator, the user, to enable removable media's data-at-rest protection according to the AGD guidance. The evaluator shall ensure that all Evaluation Activities for DAR (FDP_DAR) are conducted with the device in this configuration.

Function 21

Test 21: The evaluator shall perform the following tests.

a. The evaluator shall enable location services device-wide and shall verify that an application (such as a mapping application) is able to access the TOE's location information. The evaluator shall disable location services device-wide and shall verify that an application (such as a mapping application) is unable to access the TOE's location information.

b. [conditional] If 'per-app basis' is selected, the evaluator shall create two applications and enable one to use access the location services and the other to not access the location services. The evaluator shall exercise each application attempting to access location services individually. The evaluator shall verify that the enabled application is able to access the location services and the disabled application is not able to access the location services.

Function 22

Test 22: The evaluator shall verify that the TSS states if the TOE supports a BAF and/or hybrid authentication. If the TOE does not include a BAF and/or hybrid authentication this test is implicitly met.

a. [conditional] If a BAF is selected the evaluator shall verify that the TSS describes the procedure to enable/disable the BAF. If the TOE includes multiple BAFs, the evaluator shall verify that the TSS describes how to enable/disable each BAF, specifically if the different modalities can be individually enabled/disabled. The evaluator shall configure the TOE to allow each supported BAF to authenticate and verify that successful authentication can be achieved using the BAF. The evaluator shall configure the TOE to disable the use of each supported BAF for authentication and confirm that the BAF cannot be used to authenticate.

b. [conditional] If 'Hybrid' is selected the evaluator shall verify that the TSS describes the procedure to enable/disable the hybrid (biometric credential and PIN/password) authentication. The evaluator shall configure the TOE to allow hybrid authentication to authenticate and confirm that successful authentication can be achieved using the hybrid authentication. The evaluator shall configure the TOE to disable the use of hybrid authentication and confirm that the hybrid authentication cannot be used to authenticate.

Evaluation Activity Note: It should be noted that the following functions are optional capabilities, if the function is implemented, then the following Evaluation Activities shall be performed. The notation of ' [conditional] beside the function number indicates that if the function is not included in the ST, then there is no expectation that the evaluation activity be performed.

Function 23 [conditional]

Test 23: The test of this function is performed in conjunction with FIA_X509_EXT.2.2, FCS_TLSC_EXT.1.3 in the Package for Transport Layer Security.

Function 24 [conditional]

The evaluator shall verify that the TSS includes a list of each externally accessible hardware port and an indication of if data transfer over that port can be enabled/disabled. AGD guidance will describe how to perform the enable/disable function.

Test 24: The evaluator shall exercise the TSF configuration to enable and disable data transfer capabilities over each externally accessible hardware ports (e.g. USB, SD card, HDMI) listed by the ST author. The evaluator shall use test equipment for the particular interface to ensure that no low-level signaling is occurring on all pins used for data transfer when they are disabled. For each disabled data transfer capability, the evaluator shall repeat this test by rebooting the device into the normal operational mode and verifying that the capability is disabled throughout the boot and early execution stage of the device.

Function 25 [conditional]

The evaluator shall verify that the TSS describes how the TSF acts as a server in each of the protocols listed in the ST, and the reason for acting as a server.

Test 25: The evaluator shall attempt to disable each listed protocol in the assignment. The evaluator shall verify that remote devices can no longer access the TOE or TOE resources using any disabled protocols.

Function 26 [conditional]

Test 26: The evaluator shall exercise the TSF configuration as the administrator and, if not restricted to the administrator, the user, to enable and disable any developer mode. The evaluator shall test that developer mode access is not available when its configuration is disabled. The evaluator shall verify the developer mode remains disabled during device reboot.

Function 27 [conditional]

The evaluator shall examine the AGD guidance to determine that it describes how to enable and disable any 'Forgot Password', password hint, or remote authentication (to bypass local authentication mechanisms) capability.

Test 27: For each mechanism listed in the AGD guidance that provides a 'Forgot Password' feature or other means where the local authentication process can be bypassed, the evaluator shall disable the feature and ensure that they are not able to bypass the local authentication process.

Function 28 [conditional]

Test 28: The evaluator shall attempt to wipe Enterprise data resident on the device according to the administrator guidance. The evaluator shall verify that the data is no longer accessible by the user.

Function 29 [conditional]

The evaluator shall verify that the TSS describes how approval for an application to perform the selected action (import, removal) with respect to certificates in the Trust Anchor Database is accomplished (e.g., a pop-up, policy setting, etc.).

The evaluator shall also verify that the API documentation provided according to Section 5.2.2 Class ADV: Development includes any security functions (import, modification, or destruction of the Trust Anchor Database) allowed by applications.

Test 29: The evaluator shall perform one of the following tests:

a. [conditional] If applications may import certificates to the Trust Anchor Database, the evaluator shall write, or the developer shall provide access to, an application that imports a certificate into the Trust Anchor Database. The evaluator shall verify that the TOE requires approval before allowing the application to import the certificate:

- The evaluator shall deny the approvals to verify that the application is not able to import the certificate. Failure of import shall be tested by attempting to validate a certificate that chains to the certificate whose import was attempted (as described in the evaluation activity for FIA_X509_EXT.1).

- The evaluator shall repeat the test, allowing the approval to verify that the application is able to import the certificate and that validation occurs.

b. [conditional] If applications may remove certificates in the Trust Anchor Database, the evaluator shall write, or the developer shall provide access to, an application that removes certificates from the Trust Anchor Database. The evaluator shall verify that the TOE requires approval before allowing the application to remove the certificate:

- The evaluator shall deny the approvals to verify that the application is not able to remove the certificate. Failure of removal shall be tested by attempting to validate a certificate that chains to the certificate whose removal was attempted (as described in the evaluation activity for FIA_X509_EXT.1).

The evaluator shall repeat the test, allowing the approval to verify that the application is able to remove/modify the certificate and that validation no longer occurs.

Function 30 [conditional]

Test 30: The test of this function is performed in conjunction with FIA_X509_EXT.2.2.

Function 31 [conditional]

The evaluator shall ensure that the TSS describes which cellular protocols can be disabled. The evaluator shall confirm that the AGD guidance describes the procedure for disabling each cellular protocol identified in the TSS.

Test 31: The evaluator shall attempt to disable each cellular protocol according to the administrator guidance. The evaluator shall attempt to connect the device to a cellular network and, using network analysis tools, verify that the device does not allow negotiation of the disabled protocols.

Function 32 [conditional]

Test 32: The evaluator shall attempt to read any device audit logs according to the administrator guidance and verify that the logs may be read. This test may be performed in conjunction with the evaluation activity of FAU_GEN.1.

Function 33 [conditional]

Test 33: The test of this function is performed in conjunction with FPT_TUD_EXT.5.1.

Function 34 [conditional]

The evaluator shall verify that the TSS describes how the approval for exceptions for shared use of keys/secrets by multiple applications is accomplished (e.g., a pop-up, policy setting, etc.).

Test 34: The test of this function is performed in conjunction with FCS_STG_EXT.1.

Function 35 [conditional]

The evaluator shall verify that the TSS describes how the approval for exceptions for destruction of keys/secrets by applications that did not import the key/secret is accomplished (e.g., a pop-up, policy setting, etc.).

Test 35: The test of this function is performed in conjunction with FCS_STG_EXT.1.

Function 36 [conditional]

The evaluator shall verify that the TSS describes any restrictions in

banner settings (e.g., character limitations).

Test 36: The test of this function is performed in conjunction with FTA_TAB.1.

Function 37 [conditional]

Test 37: The test of this function is performed in conjunction with FAU_SEL.1.

Function 38 [conditional]

Test 38: The test of this function is performed in conjunction with FPT_NOT_EXT.2.1.

Function 39 [conditional]

The evaluator shall verify that the TSS includes a description of how data transfers can be managed over USB.

Test 39: The evaluator shall perform the following tests based on the selections made in the table:

a. [conditional] The evaluator shall disable USB mass storage mode, attach the device to a computer, and verify that the computer cannot mount the TOE as a drive. The evaluator shall reboot the TOE and repeat this test with other supported auxiliary boot modes.

b. [conditional] The evaluator shall disable USB data transfer without user authentication, attach the device to a computer, and verify that the TOE requires user authentication before the computer can access TOE data. The evaluator shall reboot the TOE and repeat this test with other supported auxiliary boot modes.

c. [conditional] The evaluator shall disable USB data transfer without connecting system authentication, attach the device to a computer, and verify that the TOE requires connecting system authentication before the computer can access TOE data. The evaluator shall then connect the TOE to another computer and verify that the computer cannot access TOE data. The evaluator shall then connect the TOE to the original computer and verify that the computer can access TOE data.

Function 40 [conditional]

The evaluator shall verify that the TSS includes a description of available backup methods that can be enabled/disabled. If 'selected applications or selected groups of applications are selected the TSS shall include which applications of groups of applications backup can be enabled/disabled.

Test 40: If 'all applications' is selected, the evaluator shall disable each selected backup location in turn and verify that the TOE cannot complete a backup. The evaluator shall then enable each selected backup location in turn and verify that the TOE can perform a backup.

If 'selected applications' is selected, the evaluator shall disable each selected backup location in turn and verify that for the selected application the TOE prevents backup from occurring. The evaluator shall then enable each selected backup location in turn and verify that for the selected application the TOE can perform a backup.

If 'selected groups of applications' is selected, the evaluator shall disable each selected backup location in turn and verify that for a group of applications the TOE prevents the backup from occurring. The evaluator shall then enable each selected backup location in turn and verify for the group of application the TOE can perform a backup.

If 'configuration data' is selected, the evaluator shall disable each selected backup location in turn and verify that the TOE prevents the backup of configuration data from occurring. The evaluator shall then enable each selected backup location in turn and verify that the TOE can perform a backup of configuration data.Function 41 [conditional]

The evaluator shall verify that the TSS includes a description of Hotspot functionality and USB tethering to include any authentication for these.

Test 41: The evaluator shall perform the following tests based on the selections in Function 41.

a. [conditional] The evaluator shall enable hotspot functionality with each of the of the support authentication methods. The evaluator shall connect to the hotspot with another device and verify that the hotspot functionality requires the configured authentication method.

b. [conditional] The evaluator shall enable USB tethering functionality with each of the of the support authentication methods. The evaluator shall connect to the TOE over USB with another device and verify that the tethering functionality requires the configured authentication method.

Function 42 [conditional]

Test 42: The test of this function is performed in conjunction with FDP_ACF_EXT.1.2.

Function 43 [conditional]

Test 43: The evaluator shall set a policy to cause a designated application to be placed into a particular application group. The evaluator shall then install the designated application and verify that it was placed into the correct group.

Function 44 [conditional]

Test 44: The evaluator shall attempt to unenroll the device from management and verify that the steps described in FMT_SMF_EXT.2.1 are performed. This test should be performed in conjunction with the FMT_SMF_EXT.2.1 evaluation activity.

Function 45 [conditional]

Test 45: The evaluator shall verify that the TSS contains guidance to configure the VPN as Always-On. The evaluator shall configure the VPN as Always-On and perform the following test.

a. The evaluator shall verify that when the VPN is connected all traffic is routed through the VPN. This test is performed in conjunction with FDP_IFC_EXT.1.1.

b. The evaluator shall verify that when the VPN is not established, that no traffic leaves the device. The evaluator shall ensure that the TOE has network connectivity and that the VPN is established. The evaluator shall use a packet sniffing tool to capture the traffic leaving the TOE. The evaluator shall disable the VPN connection on the server side. The evaluator shall perform actions with the device such as navigating to websites, using provided applications, and accessing other Internet resources and verify that no traffic leaves the device.

c. The evaluator shall verify that the TOE has network connectivity and that the VPN is established. The evaluator shall disable network connectivity (i.e. Airplane Mode) and verify that the VPN disconnects. The evaluator shall re-establish network connectivity and verify that the VPN automatically reconnects.

Function 46 [conditional]

Test 46: The evaluator shall verify that the TSS describes the procedure to revoke a biometric credential stored on the TOE. The evaluator shall configure the TOE to use BAF and confirm that the biometric can be used to authenticate to the device. The evaluator shall revoke the biometric credential's ability to authenticate to the TOE and confirm that the same BAF cannot be used to authenticate to the device.

Function 47

The evaluator shall verify that the TSS describes all assigned security management functions and their intended behavior.

Test 47: The evaluator shall design and perform tests to demonstrate that the function may be configured and that the intended behavior of the function is enacted by the TOE.

Function 1 – The default password limit was 6 as configured in CC Mode for testing. The evaluator demonstrated a 6-character password. See the test case for FIA_PMG_EXT.1 where the evaluator tested with the password length configured to 16 characters. The evaluator Tested password complexity and password expiration in FMT_MOF_EXT.1 test case 70, function 1.

Knox – The test was repeated on the Container password with the same results.

Function 2 – See in FMT_MOF_EXT.1 test case 70, function 2 where default timeouts are changed and the restriction is enforced. See the test case for FTA_SSL_EXT.1 where two configurable timeout values are tested. FIA_AFL_EXT.1 test case 41 and test case 42 address testing of authentication failure limits where authentication failure limits of 15 and 10 are tested.

Knox – See in FMT_MOF_EXT.1 1 test case 70, function 2 where default timeouts are changed and the restriction is enforced. See the test case for FTA_SSL_EXT.1 where two configurable timeout values are tested. FIA_AFL_EXT.1 test case 41 and test case 42 address testing of authentication failure limits.

Function 3 –

Test 72 - See the FDP_IFC_EXT.1 SFR to demonstrate the VPN works correctly. The evaluator tested the management restriction of the VPN as part of FMT_MOF_EXT.1 test case 70, function 3.

Test 73 – See FMT_MOF_EXT.1 1 test case 70, function 3 where the evaluator showed that the administrator can allow or disallow use of the VPN feature and also that the administrator can configure per-application VPN connections

Test 74 - Knox – App-specific and app-group (Knox) VPN connections were tested as part of FMT_MOF_EXT.1 1 test case 70, function 3.

Function 4 – The evaluator used a spectrum analyzer to look for NFC, Wi-Fi, Bluetooth, and cellular signals when the radios were turned off. The evaluator then turned off each protocol and ensured the signals no longer existed.

Function 5 –

Test 75- Testing for disabling of the camera and microphone was performed in FMT_MOF_EXT.1 test case 70 function 5. In regard to recovery mode, FPT_TUD_EXT.2.3 test case 103 requires the evaluator to use recovery mode. The evaluator found no method to access audio or video devices in an alternate boot mode. The evaluator attempted to gain access to either service while disabled during early-boot and auxiliary modes, however found that there was no way to access either service with or without administrator restrictions prior to the phone fully booting in its normal operational mode.

Test 76 – Not claimed.

Test 77 – Not claimed

Function 6 – The evaluator demonstrated the user can lock the device in the test for FTA_SSL_EXT.1, test case 109. The evaluator used the MDM application to lock the device as an administrator.

Function 7 – The evaluator demonstrated that a user can wipe device. The evaluator used the MDM application to wipe the device as an administrator.

Function 8 -

Test 78/79 – Application installation restrictions were tested in FMT_MOF_EXT.1 test case 70, function 8.

Knox – Container applications were tested in FMT_MOF_EXT.1 test case 70, function 8.

Function 9 – See Test Case MDFPP33:FCS_STG_EXT.1, Test 29.

Function 10 – See Test Case MDFPP33:FCS_STG_EXT.1, Test 29.

Function 11 – The evaluator used the user interface to import a client certificate and a trusted root. The evaluator then used the MDM application to import a client certificate and a trusted root. Note that the functions used for this test are the same as used for installing certificates for all of the other affected tests in the evaluation and those tests serve as evidence that certificates are installed correctly.

Function 12 – See test 11 where the evaluator successfully installed a certificate using the MDM application. The evaluator followed test 11 and used the MDM application to remove the certificate installed in Function 11 and also removed a default certificate. The evaluator repeated the test using the user interface.

Function 13 –The evaluator installed the MDM application and enabled administration. This effectively enrolls the TOE. The evaluator also installed and started the MDM application to observe the activation process. The list of MDM capabilities provided by the MDM application for management is provided prior to activation.

Function 14 – The evaluator installed an application that was used for a previous test. Once the application was installed, the evaluator proceeded to uninstall it. The evaluator checked the Application Manager to ensure it was no longer evident on the TOE. Additionally, the evaluator installed two apps and got a device file listing. The evaluator examined the device file listing and found (by searching for the apps) that both applications were present on the device. The evaluator then uninstalled two of those apps found all files and folders and subfolders related to the apps were removed during the uninstall.

Knox – The test was repeated within the container with the same results (application installed and removed).

Function 15 – See the test cases for FPT_TUD_EXT.1. However, during the course of testing no actual TOE or application updates were available. FPT_TUD_EXT.1 crafted test updates for its own purposes. Note that the various product versions used during testing were not updates but rather were fully replaced images – the version information for the overall product changed as it should in the test setup portion of this report, but none of the applications actually changed.

Function 16 – The evaluator installed applications as both a user and an administrator using the MDM.

Knox – The test was repeated within the container with the same results (applications installed).

Function 17 – See FMT_MOF_EXT.1 test case 70, function 8 where applications are installed and removed repeatedly by the user. See Function 14 where the evaluator installed and uninstalled a mobile test application as an administrator. There is no distinction between mobile and enterprise applications.

Knox – See Function 14 where Knox Container applications are addressed.

Function 18 – The evaluator locked the TOE and observed a number of notifications from previous device usage. The evaluator then configured the TOE to display no notifications when locked. The evaluator locked the device once again and found no evidence of any notifications.

Function 19 – The current version of Android is always encrypted, although it can be configured whether to require a password upon rebooting (which is required for CC Mode). This password at reboot was configured when putting the TOE in CC mode for testing.

Function 20 – See in FMT_MOF_EXT.1 test case 70, function 20 where the evaluator determined that in CC mode SD cards have to be encrypted before they can be used. They cannot be decrypted in CC Mode.

Function 21 –

Test 80 - See FMT_MOF_EXT.1 test case 70, function 21 where the evaluator tested that location services could be enabled and disabled and also blocked by an administrator.

Test 81 – Not applicable.

Function 22 –

Test 82/83 - See Test FIA_UAU.5 Test 44 where the evaluator ensured that each claimed authentication mechanism works as claimed. See also Test FIA_AFL_EXT.1 where authentication failures are tested when biometrics are disabled and then enabled.

Function 23 – See FIA_X509_EXT.2.2 and FCS_TLSC_EXT.1.3. Note that with CC enabled, invalid certificates are always rejected.

Function 24 – Not applicable.

Function 25 – Not applicable.

Function 26 – Testing in FMT_MOF_EXT.1 test case 70, function 26 demonstrates where developer mode is tested to show it can be enabled/disabled by the user and restricted/unrestricted by the administrator. Note that the tester subsequently disabled developer mode on all the TOE devices, restarted each device, and then confirmed that developer mode was still not available.

Function 27 – See FMT_MOF_EXT.1 test case 70, function 27 where the evaluator tested that the local authentication bypass can be disabled and enabled from a user and administrator perspective.

Function 28 – Testing in FMT_SMF_EXT.2 addresses where a Knox Container is wiped and where it is demonstrated that upon wiping a Knox Container all its content is removed from the filesystem.

Function 29 – Not applicable.

Function 30 – Not applicable.

Function 31 – Not applicable.

Function 32 – The evaluator dumped and read the audit records (this can be done only by an administrator).

Function 33 – Not applicable.

Function 34 – Not applicable.

Function 35 - Not applicable.

Function 36 – This was tested as part of FTA_TAB.1.

Function 37 – Not applicable.

Function 38 – Not applicable.

Function 39 –

Test case 86 - This was tested in FMT_MOF_EXT.1 test case 70, function 39 where USB Mass Storage (MTP) is tested to show it can be restricted/unrestricted by the administrator, including across reboot and in recovery mode.

Test 87 – Not applicable

Test 88 – Not applicable

Function 40 – Not applicable.

Function 41 –

Test 89/90 - This was tested in FMT_MOF_EXT.1 test case 70, function 41 where USB, Bluetooth and Wi-Fi tethering/hotspot features are tested to show they work and require the claimed authentication (PSK for Wi-Fi, implicit PSK for Bluetooth and implicit passcode for USB).

Function 42 – The test of this function is performed in conjunction with MDFPP33:FDP_ACF_EXT.1.2 where the evaluator configured the TOE to allow and disallow transfer of files between a Knox Container and the base device (Personal Mode).

Function 43 – The evaluator used the internal test application AndroidUtility and added the package name of the shared version of the application onto the App Separation list to be excluded from the separated folder inside the MDM app. The evaluator then made sure that AndroidUtility was not already installed on the device. The evaluator installed AndroidUtility and observed that the base application was installed inside a Separate Apps folder. The shared version was installed normally. The evaluator then used the applications to write to a data location previously shareable between the two versions. The evaluator then attempted to read the written data with the base app inside the separated apps container and was unable to. The evaluator concluded that the function worked as expected

Function 44 – This was tested as part of MDFPP33:FMT_SMF_EXT.2.1.

Function 45 – The Always-On VPN was tested in MDFPP33:FDP_IFC_EXT.1 test 38. Next the evaluator verified that when the VPN is not established, no traffic leaves the device. Lastly, the evaluator verified the VPN was disconnected in airplane mode.

Function 46 – Not applicable.

Function 47 – This was tested in FMT_MOF_EXT.1 test case 70, function 47 where each of these functions is tested to show the underlying function can be used and then disabled by the administrator

2.5.3 Specification of Management Functions (VPN) (VPNC24:FMT_SMF.1/VPN)

2.5.3.1 VPNC24:FMT_SMF.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check to ensure the TSS describes the client credentials and how they are used by the TOE.

Section 6.5 of the ST explains the TOE provides users the ability to specify an X.509v3 certificate (previously loaded into the TOE Platform's key store) for the TOE to use to authenticate to the VPN gateway during IPsec peer authentication.

Component Guidance Assurance Activities: The evaluator shall check to make sure that every management function mandated in the ST for this requirement is described in the operational guidance and that the description contains the information required to perform the management duties associated with each management function.

Section 3.4.1 of the Admin Guide identifies all the VPN Client settings and section 3.4.4 identifies the VPN Gateway settings.

Component Testing Assurance Activities: The evaluator shall test the TOE's ability to provide the management functions by configuring the TOE according to the operational guidance and testing each management activity listed in the ST.

The evaluator shall ensure that all management functions claimed in the ST can be performed by completing activities described in the AGD. Note that this may be performed in the course of completing other testing.

See the VPNC24:FCS_IPSEC_EXT.1 test results where VPNs using pre-shared keys and certificates are configured as needed. Each configuration identified the VPN gateway (by IP address or FQDN), client credentials (pre-shared key or certificate), reference identifier (drawn from the gateway identification), as well as any certificate used to authenticate the VPN gateway.

2.5.4 Specification of Management Functions (BT10:FMT_SMF_EXT.1/BT)

2.5.4.1 BT10:FMT_SMF_EXT.1.1/BT

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS includes a description of the Bluetooth profiles and services supported and the Bluetooth security modes and levels supported by the TOE.

If function BT-4, 'Allow/disallow additional wireless technologies to be used with Bluetooth,' is selected, the evaluator shall verify that the TSS describes any additional wireless technologies that may be used with Bluetooth, which may include Wi-Fi with Bluetooth High Speed and/or NFC as an Out of Band pairing mechanism.

If function BT-5, 'Configure allowable methods of Out of Band pairing (for BR/EDR and LE),' is selected, the evaluator shall verify that the TSS describes when Out of Band pairing methods are allowed and which ones are configurable.

If function BT-8, 'Disable/enable the Bluetooth services and/or profiles available on the OS (for BR/EDR and LE),' is selected, the evaluator shall verify that all supported Bluetooth services are listed in the TSS as manageable and, if the TOE allows disabling by application rather than by service name, that a list of services for each application is also listed.

If function BT-9, 'Specify minimum level of security for each pairing (for BR/EDR and LE),' is selected, the evaluator shall verify that the TSS describes the method by which the level of security for pairings are managed, including whether the setting is performed for each pairing or is a global setting.

Section 6.5 states that the TOE provides the management functions described in Table 9 in the ST. The table includes annotations describing the roles that have access to each service and how to access the service. The TOE enforces administrative configured restrictions by rejecting user configuration (through the UI) when attempted.

Wi-Fi Direct can be blocked to prevent Bluetooth High Speed access between devices and NFC access can be blocked to prevent out of band pairing.

Component Guidance Assurance Activities: The evaluator shall ensure that the management functions defined in the PP-Module are described in the guidance to the same extent required for the Base-PP management functions.

Section 4.3 of the Admin Guide explains how to use Bluetooth. It is the same level of detail as other management functions.

Component Testing Assurance Activities: The evaluator shall use a Bluetooth-specific protocol analyzer to perform the following tests:

Test 1: The evaluator shall disable the Discoverable mode and shall verify that other Bluetooth BR/EDR devices cannot detect the TOE. The evaluator shall use the protocol analyzer to verify that the TOE does not respond to inquiries from other devices searching for Bluetooth devices. The evaluator shall enable Discoverable mode and verify that other devices can detect the TOE and that the TOE sends response packets to inquiries from searching devices.

The following tests are conditional on if the corresponding function is included in the ST:

Test 2: (conditional): The evaluator shall examine Bluetooth traffic from the TOE to determine the current Bluetooth device name, change the Bluetooth device name, and verify that the Bluetooth traffic from the TOE lists the new name. The evaluator shall examine Bluetooth traffic from the TOE to determine the current Bluetooth device name for BR/EDR and LE. The evaluator shall change the Bluetooth device name for LE independently of the device name for BR/EDR. The evaluator shall verify that the Bluetooth traffic from the TOE lists the new name.

Test 3: (conditional): The evaluator shall disable Bluetooth BR/EDR and enable Bluetooth LE. The evaluator shall examine Bluetooth traffic from the TOE to confirm that only Bluetooth LE traffic is present. The evaluator shall repeat the test with Bluetooth BR/EDR enabled and Bluetooth LE disabled, confirming that only Bluetooth BR/EDR is present.

Test 4: (conditional): For each additional wireless technology that can be used with Bluetooth as claimed in the ST, the evaluator shall revoke Bluetooth permissions from that technology. If the set of supported wireless technologies includes Wi-Fi, the evaluator shall verify that Bluetooth High Speed is not able to send Bluetooth traffic over Wi-Fi when disabled. If the set of supported wireless technologies includes NFC, the evaluator shall verify that NFC cannot be used for pairing when disabled. For any other supported wireless technology, the evaluator shall verify that it cannot be used with Bluetooth in the specified manner when disabled. The evaluator shall then re-enable all supported wireless technologies and verify that all functionality that was previously unavailable has been restored.

Test 5: (conditional): The evaluator shall attempt to pair using each of the Out of Band pairing methods, verify that the pairing method works, iteratively disable each pairing method, and verify that the pairing method fails.

Test 6: (conditional): The evaluator shall enable Advertising for Bluetooth LE, verify that the advertisements are captured by the protocol analyzer, disable Advertising, and verify that no advertisements from the device are captured by the protocol analyzer.

Test 7: (conditional): The evaluator shall enable Connectable mode and verify that other Bluetooth devices may pair with the TOE and (if the devices were bonded) reconnect after pairing and disconnection. For BR/EDR devices: The evaluator shall use the protocol analyzer to verify that the TOE responds to pages from the other devices and permits pairing and re-connection. The evaluator shall disable Connectable mode and verify that the TOE does not respond to pages from remote Bluetooth devices, thereby not permitting pairing or re-connection. For LE: The evaluator shall use the protocol analyzer to verify that the TOE sends connectable advertising events and responds to connection requests. The evaluator shall disable Connectable mode and verify that the TOE stops sending connectable advertising events and stops responding to connection requests from remote Bluetooth devices.

Test 8: (conditional): For each supported Bluetooth service and/or profile listed in the TSS, the evaluator shall verify that the service or profile is manageable. If this is configurable by application rather than by service and/or profile name, the evaluator shall verify that a list of services and/or profiles for each application is also listed.

Test 9: (conditional): The evaluator shall allow low security modes/levels on the TOE and shall initiate pairing with the TOE from a remote device that allows only something other than Security Mode 4/Level 3 or Security Mode 4/Level 4 (for BR/EDR), or Security Mode 1/Level 3 (for LE). (For example, a remote BR/EDR device may claim Input/Output capability 'NoInputNoOutput' and state that man-in-the- middle (MiTM) protection is not required. A remote LE device may not support encryption.) The evaluator shall verify that this pairing attempt succeeds due to the TOE falling back to the low security mode/level. The evaluator shall then remove the pairing of the two devices, prohibit the use of low security modes/levels on the TOE, then attempt the connection again. The evaluator shall verify that the pairing attempt fails. With the low security modes/levels disabled, the evaluator shall initiate pairing from the TOE to a remote device that supports Security Mode 4/Level 3 or Security Mode 4/Level 4 (for BR/EDR) or Security Mode 1/Level 3 (for LE). The evaluator shall verify that this pairing is successful and uses the high security mode/level.

Test 1 – The evaluator first turned-on discoverable mode and demonstrated the device could be seen by other devices. The evaluator then disabled discovery mode and attempted to scan once again and found that no TOE devices could be discovered and the UI reported that discoverable mode was not allowed.

Test 2 – Not applicable.

Test 3 – Not applicable.

Test 4 – See Test Case FMT_MOF_EXT.1 test case 70, function 4 where the evaluator disables NFC and ensures that an NFC connection cannot be made and Test Case FMT_SMF_EXT.1 function 4 where the evaluator ensures there are no RF signals when disabled.

Test 5 – Not applicable.

Test 6 – Not applicable.

Test 7 – Not applicable.

Test 8 – Not applicable.

Test 9 – Not applicable.

2.5.5 Specification of Management Functions (WLAN Client) - per TD0667 (WLANC10:FMT_SMF_EXT.1/WLAN)

2.5.5.1 WLANC10:FMT_SMF_EXT.1.1/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall check to make sure that every management function mandated by the EP is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.

The embedded spreadsheet in Section 3.3 of the Admin Guide explains how to configure wireless networks settings.

Component Testing Assurance Activities: The evaluator shall test the TOE's ability to provide the management functions by configuring the TOE and testing each option listed in the requirement above.

Note that the testing here may be accomplished in conjunction with the testing of other requirements, such as FCS_TLSC_EXT and FTA_WSE_EXT.

Test -

FMT_SMF_EXT.1.1/WLAN WL1 – The evaluator tested that a security policy can be specified for a wireless network and that only correct connections can connect from both a user and administrator perspective.

FMT_SMF_EXT.1.1/WLAN WL2 – The evaluator demonstrated the user could connect to networks using SSIDs. The MDM was then used to configure SSIDs and restrict access to certain SSIDs. The restrictions were enforced when in place and once removed, the user could freely connect to networks.

2.5.6 Specification of Remediation Actions (MDFPP33:FMT_SMF_EXT.2)

2.5.6.1 MDFPP33:FMT_SMF_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes all available remediation actions, when they are available for use, and any other administrator-configured triggers. The evaluator shall verify that the TSS describes how the remediation actions are provided to the administrator.

Section 6.5 of the ST explains user can unenroll an MDM agent from the device in one of two ways. First, a user can revoke the MDM agent's administrative privileges through the Settings app (Settings->Security & Location->Device admin apps) and then uninstall the agent. This method assumes that the MDM agent does not block the user from revoking the agent's administrator privileges. When unenrolled in this fashion, the device will remove the work profile, the work profile's applications and data. In effect, this translates to the selections of wiping all sensitive data (all work profile data), removing all Enterprise applications (all work profile applications), removing all device-stored Enterprise application data (all work profile application data), and removing Enterprise secondary authentication data (Knox password and/or fingerprint).

In the case where an MDM agent blocks the user from revoking the agent's administrative privileges, a user can only unenroll by wiping the entire device. By doing this, the user causes the device to wipe all protected data (wiping all data, including both the user's protected data and any Enterprise/work profile data) as well as remove MDM policies and disabling CC mode (as the device returns to factory defaults).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall use the test environment to iteratively configure the device to perform each remediation action in the selection. The evaluator shall configure the remediation action per how the TSS states it is provided to the administrator. The test environment could be a MDM agent application, but can also be an application with administrator access.

Test 1 – See Test Case FMT_MOF_EXT.1 test case 70, function 47 - enable/disable Admin removal where the user unenrolled the TOE. Once the device was unenrolled, the evaluator could no longer perform administrator functions and the Knox container previously created was automatically removed.

In the case of wiping the device in order to unenroll, see Test Case FMT_SMF_EXT.1 function 7 where the evaluator shows the device can be wiped by an administrator and there are several other tests in this report showing that the user can wipe the device (FCS_CKM_EXT.5) or the device will be wiped as a result of successively entering incorrect passwords (FIA_AFL_EXT.1). In each case that the evaluator wiped the device, the device was reset and had to go through initial setup by the evaluator. The evaluator then had to install the MDM app and re-enroll the device, following the procedures outlined in the base test report, in order to get it back into CC mode to continue testing. As such, the evaluator has determined that wiping the device results in an unenrollment with the side effect of wiping the device as claimed.

In the case of wiping sensitive data, there is no specific function to do that, but performing a wipe of protected data also results in a wipe of sensitive data.

In the cases of both wiping the device and removing Enterprise apps, if an MDM were to unenroll a mobile device it can certainly instruct its agent(s) to perform any available administrator functions immediately prior to the final unenrollment or incidentally resulting in unenrollment (in the case of instructing a wipe). As such FMT_SMF_EXT.1 function 7 serves to show that the device (protected and sensitive data) could be wiped and enterprise applications can be removed as a result of unenrollment as claimed. Those same test cases address Knox containers as, in wiping the TOE device, all Knox Containers are wiped along with the rest of the device contents.

2.5.7 Current Administrator (MDFPP33:FMT_SMF_EXT.3)

2.5.7.1 MDFPP33:FMT_SMF_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall cause the TOE to be enrolled into management. The evaluator shall then invoke this mechanism and verify the ability to view that the device has been enrolled, view the management functions that the administrator is authorized to perform.

Test – The evaluator enrolled the TOE into management. The evaluator examined the device settings for Device Admin Apps to first identify the enrolled application and then to review the permissions assigned to that app.

2.6 Protection of the TSF (FPT)

2.6.1 Application Address Space Layout Randomization (MDFPP33:FPT_AEX_EXT.1)

2.6.1.1 MDFPP33:FPT_AEX_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.1.2 MDFPP33:FPT_AEX_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS section of the ST describes how the 8 bits are generated and provides a justification as to why those bits are unpredictable.

Section 6.6 of the ST states the Linux kernel of the TOE's Android operating system provides address space layout randomization utilizing a non-cryptographic kernel random function to provide 8 unpredictable bits to the base address of any user-space memory mapping. The random function, though not cryptographic, ensures that one cannot predict the value of the bits.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation Activity Note: The following test require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

Test 94: The evaluator shall select 3 apps included with the TSF. These must include any web browser or mail client included with the TSF. For each of these apps, the evaluator shall launch the same app on two separate Mobile Devices of the same type and compare all memory mapping locations. The evaluator must ensure that no memory mappings are placed in the same location on both devices.

If the rare (at most 1/256) chance occurs that two mappings are the same for a single app and not the same for the other two apps, the evaluator shall repeat the test with that app to verify that in the second test the mappings are different.

Test 94 – The evaluator had 2 devices of each type to perform this test. The evaluator chose the email, settings, and chrome applications to perform this test. The evaluator started the matching apps on each device and performed a memory map of each device. The evaluator then used a test program to compare the memory locations of each app on each device. The mapping for each app was different on each device.

2.6.2 Memory Page Permissions (MDFPP33:FPT_AEX_EXT.2)

2.6.2.1 MDFPP33:FPT_AEX_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS describes of the memory management unit (MMU), and ensures that this description documents the ability of the MMU to enforce read, write, and execute permissions on all pages of virtual memory.

Section 6.6 of the ST states the Android 14 operating system utilizes 6.1/5.10/4.19 Linux kernels, whose memory management unit (MMU) enforces read, write, and execute permissions on all pages of virtual memory and ensures that write and execute permissions are not simultaneously granted on all memory (exceptions are only made for Dalvik JIT compilation). The Android operating system sets the ARM eXecute Never (XN) bit on memory pages and the MMU circuitry of the TOE's ARMv8 Application Processor enforces the XN bits. From Android's security documentation (https://source.android.com/security/), Android supports "Hardware-based No eXecute (NX) to prevent code execution on the stack and heap". Section D.5 of the ARM v8 Architecture Reference Manual contains additional details about the MMU of ARM-based processors: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0487a.f/index.html.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.3 Stack Overflow Protection (MDFPP33:FPT_AEX_EXT.3)

2.6.3.1 MDFPP33:FPT_AEX_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall determine that the TSS contains a description of stackbased buffer overflow protections implemented in the TSF software which runs in the non-privileged execution mode of the application processor. The exact implementation of stack-based buffer overflow protection will vary by platform. Example implementations may be activated through compiler options such as '-fstack-protector-all', '-fstack-protector', and '/GS' flags. The evaluator shall ensure that the TSS contains an inventory of TSF binaries and libraries, indicating those that implement stack-based buffer overflow protections as well as those that do not. The TSS must provide a rationale for those binaries and libraries that are not protected in this manner.

Section 6.6 of the ST states the TOE's Android operating system provides explicit mechanisms to prevent stack buffer overruns (enabling -fstack-protector) in addition to taking advantage of hardware-based No eXecute to prevent code execution on the stack and heap. Samsung requires and applies these protections to all TSF executable binaries and libraries.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.4 Domain Isolation (MDFPP33:FPT_AEX_EXT.4)

2.6.4.1 MDFPP33:FPT_AEX_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.4.2 MDFPP33:FPT_AEX_EXT.4.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS describes the mechanisms that are in place that prevents non-TSF software from modifying the TSF software or TSF data that governs the behavior of the TSF. These mechanisms could range from hardware-based means (e.g. 'execution rings' and memory management functionality); to software-based means (e.g. boundary checking of inputs to APIs). The evaluator determines that the described mechanisms appear reasonable to protect the TSF from modification.

The evaluator shall ensure the TSS describes how the TSF ensures that the address spaces of applications are kept separate from one another.

The evaluator shall ensure the TSS details the USSD and MMI codes available from the dialer at the locked state or during auxiliary boot modes that may alter the behavior of the TSF. The evaluator shall ensure that this description includes the code, the action performed by the TSF, and a justification that the actions performed do not modify user or TSF data. If no USSD or MMI codes are available, the evaluator shall ensure that the TSS provides a description of the method by which actions prescribed by these codes are prevented.

The evaluator shall ensure the TSS documents any TSF data (including software, execution context, configuration information, and audit logs) which may be accessed and modified over a wired interface in auxiliary boot modes. The evaluator shall ensure that the description includes data, which is modified in support of update or restore of the device. The evaluator shall ensure that this documentation includes the auxiliary boot modes in which the data may be modified, the methods for entering the auxiliary boot modes, the location of the data, the manner in which data may be modified, the data format and packaging necessary to support modification, and software and/or hardware tools, if any, which are necessary for modifying the data.

The evaluator shall ensure that the TSS provides a description of the means by which unauthorized and undetected modification (that is, excluding cryptographically verified updates per FPT_TUD_EXT.2) of the TSF data over the wired interface in auxiliary boots modes is prevented. The lack of publicly available tools is not sufficient justification. Examples of sufficient justification include auditing of changes, cryptographic verification in the form of a digital signature or hash, disabling the auxiliary boot modes, and access control mechanisms that prevent writing to files or flashing partitions.

Section 6.6 of the ST describes TOE protects itself from modification by untrusted subjects using a variety of methods. The first protection employed by the TOE is a Secure Boot process that uses cryptographic signatures to ensure the authenticity and integrity of the bootloader and kernels using data fused into the device processor.

The TOE's Secure Boot process employs a series of public keys to form a chain of trust that operates as follows. The Application Processor (AP) contains the hash of the Secure Boot Public Key (a key embedded in the end of the signed bootloader image), and upon verifying the SBPK attached to the bootloader produces the expected hash, the AP uses this public key to verify the signature of the bootloader image, to ensure its integrity and authenticity before transitioning execution to the bootloader. The bootloader, in turn, contains the Image Signing Public Key (ISPK), which the bootloader will use to verify the signature on either kernel image (primary kernel image or recovery kernel image). The signing key type and hash type used are listed in Table 19 - Secure Boot Public Keys.

Device	Signing Key	Hash
All other devices	ECDSA P384	SHA-256
Galaxy Z Flip6 5G	RSA 2048	SHA-256

Table 19 - Secure Boot Public Keys

Note that when configured for Common Criteria mode, the TOE only accepts updates to the TOE via FOTA; however, when not configured for CC mode, the TOE allows updates through the bootloader's ODIN mode. The primary kernel includes an embedded FOTA Public Key, which the TOE uses to verify the authenticity and integrity of FOTA update signatures (which contain a PKCS 2.1 PSS RSA 2048 w/ SHA-256 signature).

The TOE protects access to the REK and derived HEK to only trusted applications2 within the TEE (TrustZone). The TOE key manager includes a TEE module that utilizes the HEK to protect all other keys in the key hierarchy. All TEE applications are cryptographically signed, and when invoked at runtime (at the behest of an untrusted application), the TEE will only load the trusted application after successfully verifying its cryptographic signature. Furthermore, the device encryption library checks the integrity of the system by checking the result from both Secure Boot/SecurityManager and from the Integrity Check Daemon before servicing any requests. Without this TEE application, no keys within the TOE (including keys for ScreenLock, the keystore, and user data) can be successfully decrypted, and thus are useless.

The TOE protects biometric data by separating it from the Android operating system. The biometric sensor is tied to the TEE such that it cannot be accessed directly from Android but can only be done through the biometric software inside the TEE. All biometric data is maintained within the TEE such that Android is only able to know the result of a biometric process (such as enrollment or verification), and not any of the data used in that process itself.

[2] A TrustZone application is a trusted application that executes in a hardware-isolated domain.

The third protection is the TOE's internal SecurityManager watchdog service. The SecurityManager manages the CC mode of the TOE by looking for unsigned kernels or failures from other, non-cryptographic checks on system integrity, and upon detection of a failure in either, disables the CC mode and notifies the TEE application. The TEE application then locks itself, again rendering all TOE keys useless.

Finally, the TOE's Android OS provides "sandboxing" that ensures each non-system mobile application executes with the file permissions of a unique Linux user ID in a different virtual memory space. This ensures that applications cannot access each other's memory space (it is possible for two processes to utilize shared memory, but not directly access the memory of another application) or files and cannot access the memory space or files of system-level applications.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation Activity Note: The following tests require the vendor to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products. In addition, the vendor provides a list of files (e.g., system files, libraries, configuration files, audit logs) that make up the TSF data. This list could be organized by folders/directories (e.g., /usr/sbin, /etc), as well as individual files that may exist outside of the identified directories.

Test 95: The evaluator shall create and load an app onto the Mobile Device. This app shall attempt to traverse over all file systems and report any locations to which data can be written or overwritten. The evaluator must ensure that none of these locations are part of the OS software, device drivers, system and security configuration files, key material, or another untrusted application's image/data. For example, it is acceptable for a trusted photo editor app to have access to the data created by the camera app, but a calculator application shall not have access to the pictures.

Test 96: For each available auxiliary boot mode, the evaluator shall attempt to modify a TSF file of their choosing using the software and/or hardware tools described in the TSS. The evaluator shall verify that the modification fails.

Test 95 – The evaluator loaded a developer provided application and examined partitions. The evaluator attempted to create a file in a restricted partition and could not write in that partition as that is where the system files are stored. The evaluator then successfully created a file in the partition which is writable by users to demonstrate files can be created. The previous test case attempted to modify a system file and failed. Using a final file dump created by the evaluator, the evaluator searched the system partition for writeable locations/files – none were found. The evaluator then searched for the installed apps – checked to see if apps had unique UIDs/GIDs – they either had system/system, special (such as radio), or unique UIDs/GIDs. It is assumed that system/special things are pre-installed functional apps. None of the pre-installed apps can be removed, only disabled. The evaluator then looked for the installed app UID&GID and found them only attached with associated files. Next the evaluator looked at the app folders and found they were all protected using the UID/GID of the app so no other apps would have access.

Test 96 – The developer provided a program and script to use the program to reboot the device into recovery mode and then traverse the filesystem attempting to delete and modify files across the filesystem, including TSF files. The evaluator was unable to modify a TSF file.

2.6.5 Kernel Address Space Layout Randomization (MDFPP33:FPT_AEX_EXT.5)

2.6.5.1 MDFPP33:FPT_AEX_EXT.5.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.5.2 MDFPP33:FPT_AEX_EXT.5.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS section of the ST describes how the bits are generated and provides a justification as to why those bits are unpredictable.

Section 6.6 of the ST explains the TOE provides Kernel Address Space Layout Randomization to ensure that the base address of kernel-space memory mappings consist of six (6) unpredictable bits. This ensures that at each boot, the location of kernel data structures including the core kernel begins at a random physical address, mapping the core kernel at a random virtual address in the vmalloc area, loading kernel modules at a random virtual address in the vmalloc area, and mapping system memory at a random virtual address in the linear area.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation Activity Note: The following test require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

Test 133: The evaluator shall reboot the TOE six times. For each of these reboots, the evaluator shall examine memory mapping locations of the kernel. The evaluator must ensure that for at least five reboots the memory mappings are not placed in the same location on both devices.

Test 133 – The evaluator had 2 devices of each type to perform this test. The evaluator rebooted the devices 5 times each and took a memory map of the kernel. The evaluator then compared those maps to ensure the kernel was not loaded into the same location on a device

2.6.6 Write or Execute Memory Page Permissions (MDFPP33:FPT_AEX_EXT.6)

2.6.6.1 MDFPP33:FPT_AEX_EXT.6.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS describes how the operating system of the application processor prevents all processes executing in a non-privileged execution domain from achieving write and execute permissions on any page of memory (with only specified exceptions). The evaluator shall ensure that the TSS describes how such processes are unable to request pages of memory with such permissions, and how they are unable to change permissions to both write and execute on any pages already allocated to them.

Section 6.6 of the ST states the TOE's Android 14 operating system utilizes 6.1/5.10/4.19 Linux kernels, whose memory management unit (MMU) enforces read, write, and execute permissions on all pages of virtual memory and ensures that write and execute permissions are not simultaneously granted on all memory (exceptions are only made for Dalvik JIT compilation). The Android operating system sets the ARM eXecute Never (XN) bit on memory pages and the MMU circuitry of the TOE's ARMv8 Application Processor enforces the XN bits. From Android's security documentation (https://source.android.com/security/), Android supports "Hardware-based No eXecute (NX) to prevent code execution on the stack and heap". Section D.5 of the ARM v8 Architecture Reference Manual contains additional details about the MMU of ARM-based processors:
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0487a.f/index.html.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.7 Application Processor Mediation (MDFPP33:FPT_BBD_EXT.1)

2.6.7.1 MDFPP33:FPT_BBD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS section of the ST describes at a high level how the processors on the Mobile Device interact, including which bus protocols they use to communicate, any other devices operating on that bus (peripherals and sensors), and identification of any shared resources. The evaluator shall verify that the design described in the TSS does not permit any BPs from accessing any of the peripherals and sensors or from accessing main memory (volatile and non-volatile) used by the AP. In particular, the evaluator shall ensure that the design prevents modification of executable memory of the AP by the BP.

Section 6.6 of the ST explains the TOE's hardware and software architecture ensures separation of the application processor (AP) from the baseband or communications processor (CP). While the AP and CP are part of the same SoC, they are separate physical components within the SoC with no shared components between them (such as memory) and communicate via a non-shared bus (i.e. no other chips can communicate via this bus). From a software perspective, the AP and CP communicate logically through the Android Radio Interface Layer (RIL) daemon. This daemon, which executes on the AP, coordinates all communication between the AP and CP. It makes requests of the CP and accepts the response from the CP; however, the RIL daemon does not provide any reciprocal mechanism for the CP to make requests of the AP. Because the mobile architecture provides only the RIL daemon interface, the CP has no method to access the resources of the software executing on the AP.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.8 Biometric data processing (BIO11:FPT_BDP_EXT.1)

2.6.8.1 BIO11:FPT_BDP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.8.2 BIO11:FPT_BDP_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: As depicted in Figure 1 of [BIOPP-Module], biometric characteristics are captured by a biometric capture sensor and then sent to the processors in the computer for signal processing, PAD and comparison and the decision outcome is returned. This is a typical process flow of biometric verification; however, a biometric capture sensor may do all the tasks within the sensor. In either case, all TSF modules (i.e. biometric capture sensor and any software running in biometric capture sensor and the computer processors) that process plaintext biometric data must be separated from any entities outside the SEE. Any plaintext biometric data must not be accessible from any entities outside the SEE.

In any case, the evaluator shall examine the TSS to confirm that;

a. All TSF modules and physical interconnections are within the defined boundary of the SEE and any entities outside the SEE including the main computer operating system can't interfere with transmission between and processing of these modules

b. All plaintext biometric data (whether generated by the biometric capture sensor or by the evaluation processes of the TSF) is retained in volatile memory within the SEE and any entities outside the SEE including the main computer operating system can't access these data. Any TSFIs which may exist, do not reveal plaintext biometric data to any entities outside the SEE. The evaluator shall examine TSFIs of TSF modules provided by the biometric capture sensor (e.g. SDK) because they may include testing or debug codes and the developer who integrated the sensor into the TOE may apply changes to those modules

The evaluator shall keep in mind that the objective of this EA is not evaluating the SEE itself. This EA is derived from ASE_TSS.1.1 which requires that the TSS and BMD to provide potential consumers of the TOE with a high-level view of how the developer intends to satisfy each SFR. The evaluator shall check the TSS and BMD to seek for a logical explanation how the above criteria are satisfied considering this scope of the requirement.

Pass/Fail criteria

The evaluator can pass this EA only if the evaluator confirms that:

a. Information necessary to perform this EA is described in the TSS and BMD

Requirements for reporting

The evaluator shall report the summary of result of EA defined above, especially how the evaluator reaches the Pass/Fail judgement based on the Pass/Fail criteria.

Section 6.6 of the TSS states the complete biometric authentication process happens inside the TEE (including image capture, all processing and match determination). All software in the biometric system is inside the TEE boundary, while the sensors are accessible from within Android. The TEE handles calls for authentication made from Android with only the success or failure of the match provided back to Android (and when applicable, to the calling app). The image taken by the capture sensor is processed by the biometric service to check the enrolled templates for a match to the captured image. The KMD provides additional details to support this requirement.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Plaintext biometric data must not be accessible from any entities outside the SEE, especially the main computer operating system, to meet FPT_BDP_EXT.1. This means that:

a. The TOE must not expose the plain biometric data to the memory that is accessible by the main computer operating system during the processing of biometric data.

b. Any TSFIs identified in the TSS that can output plaintext biometric data must not be accessible by the main computer operating system for local storage.

From a testing perspective, a) is a function of ensuring the TOE design such that it is contained within the SEE boundaries. As the SEE itself is out of scope, there is no specific test to verify this as it brings the SEE itself into the scope of the testing (as if the TOE design is correct any failure would be due to an SEE failure).

If the TSS describes any TSFIs that may export plaintext biometric data (such as those provided by an SDK for integration and debugging), the evaluator must verify that b) is true.

If it is impractical or inadequate to conduct the following tests, the developer may propose alternate approaches to verify them. It is the evaluator's responsibility to determine the suitability of an alternate approach. For example, an analysis of source code to determine that they are met by the TOE is an acceptable alternate approach, as described in the [CEM].

a. TSFI invocation test

If TSFIs exist, they could be used to output plaintext biometric data to the operating system, and so the evaluator shall perform this test. The following test steps require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

1. The evaluator shall identify any TSFIs that output plaintext biometric data to the memory that is accessible by the operating system.

2. The evaluator shall attempt to access and TSFIs using the developer provided test platform and tools to verify that no plaintext data can be accessed from the main computer operating system.

Pass/Fail criteria

The evaluator can pass this EA only if the evaluator confirms that:

a. Information necessary to perform this EA is described in the TSS and BMD

b. The TOE does not provide access to the biometric transaction to the main computer operating system

c. No TSFIs provide access to plaintext biometric data to the main computer operating system

Requirements for reporting

The evaluator shall report the summary of result of EA defined above, especially how the evaluator reaches the Pass/Fail judgement based on the Pass/Fail criteria.

Not applicable as there are no TOE Security Functions Interfaces (TSFIs) that can output plaintext biometric data.

2.6.9 JTAG Disablement (MDFPP33:FPT_JTA_EXT.1)

2.6.9.1 MDFPP33:FPT_JTA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If disable access through hardware is selected:

The evaluator shall examine the TSS to determine the location of the JTAG ports on the TSF, to include the order of the ports (i.e. Data In, Data Out, Clock, etc.).

If control access by a signing key is selected:

The evaluator shall examine the TSS to determine how access to the JTAG is controlled by a signing key. The evaluator shall examine the TSS to determine when the JTAG can be accessed, i.e. what has the access to the signing key.

Section 6.6 of the ST states the TOE prevents access to its processor’s JTAG interface by only enabling JTAG when the TOE has a special image written to its bootloader/TEE partitions. That special image must be signed by the appropriate key (corresponding to the public key that has its SHA-256 hash programmed into the processor’s fuses).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation Activity Note: The following test requires the developer to provide access to a test platform that provides the evaluator with chip level access.

If 'disable access through hardware' is selected:

The evaluator shall connect a packet analyzer to the JTAG ports. The evaluator shall query the JTAG port for its device ID and confirm that the device ID cannot be retrieved.

Test – The test is not applicable as the ST selected 'control access by a signing key' in the requirement.

2.6.10 Key Storage (MDFPP33:FPT_KST_EXT.1)

2.6.10.1 MDFPP33:FPT_KST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall consult the TSS section of the ST in performing the Evaluation Activities for this requirement.

In performing their review, the evaluator shall determine that the TSS contains a description of the activities that happen on power-up and password authentication relating to the decryption of DEKs, stored keys, and data.

The evaluator shall ensure that the description also covers how the cryptographic functions in the FCS requirements are being used to perform the encryption functions, including how the KEKs, DEKs, and stored keys are unwrapped, saved, and used by the TOE so as to prevent plaintext from being written to non-volatile storage. The evaluator shall ensure that the TSS describes, for each power-down scenario how the TOE ensures that all keys in non-volatile storage are not stored in plaintext.

The evaluator shall ensure that the TSS describes how other functions available in the system (e.g., regeneration of the keys) ensure that no unencrypted key material is present in persistent storage.

The evaluator shall review the TSS to determine that it makes a case that key material is not written unencrypted to the persistent storage.

Section 6.6 of the ST states the TOE does not store any plaintext key or biometrics material in its internal Flash; instead, the TOE encrypts all keys and biometric data before storing them. This ensures that irrespective of how the TOE powers down (e.g., a user commands the TOE to power down, the TOE reboots itself, or battery is removed), all keys and biometric data stored in internal Flash are wrapped with a KEK. Please refer to section 6.2 of the TSS for further information (including the KEK used) regarding the encryption of keys stored in the internal Flash. Additionally, the KMD provides details with respect to key protection. As the TOE encrypts all keys stored in Flash, upon boot-up the TOE must first decrypt and utilize keys. Note as well that the TOE does not use a user's biometric fingerprint to encrypt/protect key material. Rather the TOE always requires the user enter his or her password after a reboot (in order to derive the necessary keys to decrypt the user data partition and other keys in the key hierarchy).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.11 No Key Transmission (MDFPP33:FPT_KST_EXT.2)

2.6.11.1 MDFPP33:FPT_KST_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall consult the TSS section of the ST in performing the Evaluation Activities for this requirement. The evaluator shall ensure that the TSS describes the TOE security boundary. The cryptographic module may very well be a particular kernel module, the Operating System, the Application Processor, or up to the entire Mobile Device.

In performing their review, the evaluator shall determine that the TSS contains a description of the activities that happen on power-up and password authentication relating to the decryption of DEKs, stored keys, and data.

The evaluator shall ensure that the TSS describes how other functions available in the system (e.g., regeneration of the keys) ensure that no unencrypted key material is transmitted outside the security boundary of the TOE.

The evaluator shall review the TSS to determine that it makes a case that key material is not transmitted outside the security boundary of the TOE.

Section 6.6 of the ST states that the TOE utilizes a cryptographic library consisting of an implementation of BoringSSL, the Kernel Crypto module, the SCrypto module, and the following system-level executables that utilize KEKs: fscrypt (on devices with FBE), eCryptfs, wpa_supplicant, and the keystore.

The TOE ensures that plaintext key material is not exported by not allowing the REK to be exported and by ensuring that only authenticated entities can request utilization of the REK. Furthermore, the TOE only allows the system-level executables access to plaintext DEK values needed for their operation. The TSF software (the system-level executables) protects those plaintext DEK values in memory both by not providing any access to these values and by clearing them when no longer needed (in compliance with FCS_CKM_EXT.4). Note again that the TOE does not use the user’s biometric fingerprint to encrypt/protect key material (and instead only relies upon the user’s password).

The TOE also ensures that biometric data used for enrolling and authenticating users cannot be exported. During authentication or enrollment, the calling program (the TSF or an app) is able to request biometric actions, but the data resulting from that action is not provided back to the calling program. The calling program only receives a notice of success of failure about the process.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.12 No Plaintext Key Export (MDFPP33:FPT_KST_EXT.3)

2.6.12.1 MDFPP33:FPT_KST_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The ST author will provide a statement of their policy for handling and protecting keys. The evaluator shall check to ensure the TSS describes a policy in line with not exporting either plaintext DEKs, KEKs, or keys stored in the secure key storage.

Section 6.6 of the ST states the TOE does not provide any way to export plaintext DEKs or KEKs (including all keys stored in the keystore) as the TOE chains all KEKs to the HEK/REK. The evaluator finds this statement consistent with the rest of the TSS which does not discuss exporting plaintext keys. This conclusion is also supported by material in the KMD.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.13 Self-Test Notification (MDFPP33:FPT_NOT_EXT.1)

2.6.13.1 MDFPP33:FPT_NOT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes critical failures that may occur and the actions to be taken upon these critical failures.

Section 6.6 of the ST identifies 2 critical failures - a self-test failure or TOE software integrity verification failure. If one of these critical failures occurs, the TOE transitions to a non-operational mode. The user may attempt to power-cycle the TOE to see if the failure condition persists, and if it does persist, the user may attempt to boot to the recovery mode/kernel to wipe data and perform a factory reset in order to recover the device.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation Activity Note: The following test require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

Test 97: The evaluator shall use a tool provided by the developer to modify files and processes in the system that correspond to critical failures specified in the second list. The evaluator shall verify that creating these critical failures causes the device to take the remediation actions specified in the first list.

Test 97 – The evaluator ran two test cases. In the first case an algorithm self-test failed (because the vendor had provided a bad algorithm implementation to use for this testing). In the second case, the evaluator attempted to load a custom kernel. In all cases, the TOE needed to be reloaded.

2.6.14 Protection of biometric template - per TD0714 (BIO11:FPT_PBT_EXT.1)

2.6.14.1 BIO11:FPT_PBT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS and AGD guidance to identify any TSFI through which the user can access (e.g. revoke) the templates. The evaluator shall confirm that those TSFI requires using a NBAF as specified by the ST author.

Section 6.6 of the ST states the TOE requires the user to enter their password to enroll, re-enroll or unenroll any biometric templates. When the user attempts biometric authentication to the TOE, the biometric sensor takes an image of the presented biometric for comparison to the enrolled templates. The biometric system compares the captured image to all the stored templates on the device to determine if there is a match.

Component Guidance Assurance Activities: The evaluator shall examine the TSS and AGD guidance to identify any TSFI through which the user can access (e.g. revoke) the templates. The evaluator shall confirm that those TSFI requires using a NBAF as specified by the ST author.

Section 2.5.1 of the Admin Guide states to enroll a fingerprint, the user must first establish a password for the device. Once set, users can navigate to the device's settings to manage fingerprints, add new fingerprints, or delete existing ones. However, to perform these actions, users must enter their password.

When enrolling a fingerprint, the device provides on-screen guidance to capture quality data, asking users to move their finger to cover a wider area. To acquire optimal fingerprint samples, the device prompts users to ensure they maintain proper finger placement on the sensor, providing multiple user-friendly messages until quality data acquisition is complete.

It is important to note that biometric authentication cannot be used immediately following a power event, such as a power-on or a reboot. Users must first authenticate using their password before gaining access to biometric authentication. Once authenticated, users can utilize their fingerprints for up to 72 hours without requiring a device reboot.

Component Testing Assurance Activities: The evaluator shall perform the following test steps to verify that the TOE protects the templates as specified in TSS and AGD guidance.

1. The evaluator shall perform functions through the TSFIs that access the templates

2. The evaluator shall check that the TSFI requires using a NBAF as specified by the ST author

a. The evaluator shall attempt to add new, change and remove existing BAFs on the TOE and ensure the NBAF is required before allowing any change

See Test Case FIA_UAU.6/CREDENTIAL test case 47 where the evaluator showed that the NBAF (password) is required before allowing any changes to the existing BAFs (fingerprint) on the TOE.

2.6.15 Reliable time stamps (MDFPP33:FPT_STM.1)

2.6.15.1 MDFPP33:FPT_STM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it lists each security function that makes use of time. The TSS provides a description of how the time is maintained and considered reliable in the context of each of the time related functions. This documentation must identify whether the TSF uses a NTP server or the carrier's network time as the primary time sources.

Section 6.6 of the ST identifies the following security functions that make use of time: Package Manager, FOTA image verifier, TLS certificate validation, wpa_supplicant, audit system and keystore applications. These TOE components obtain time from the TOE using system API calls [e.g., time() or gettimeofday()]. An application cannot modify the system time as mobile applications need the Android “SET_TIME” permission to do so. Likewise, only a process with system privileges can directly modify the system time using system-level APIs. The TOE uses the Cellular Carrier time (obtained through the Carrier’s network timeserver) as a trusted source; however, the user can also manually set the time through the TOE’s user interface.

Component Guidance Assurance Activities: The evaluator examines the operational guidance to ensure it describes how to set the time.

The embedded spreadsheet in Section 3.3 of the Admin Guide provides the MDM interface, setAutomaticTime(), for making time solely reliant on the local device.

Component Testing Assurance Activities: Test 98: The evaluator uses the operational guide to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

Test 98 – The evaluator turned off Wi-Fi to ensure the clock would not automatically update and examined the current time on the TOE by pressing down on the time in order to also see the date. The evaluator then changed the time and date by unchecking the automatic date/time setting and manually changing the time. The evaluator then re-enabled the automatic date and time setting and the Wi-Fi to cause the time to be reset. The evaluator observed the restored date and time.

2.6.16 TSF Cryptographic Functionality Testing (MDFPP33:FPT_TST_EXT.1)

2.6.16.1 MDFPP33:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it specifies the self-tests that are performed at start-up. This description must include an outline of the test procedures conducted by the TSF (e.g., rather than saying 'memory is tested', a description similar to 'memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written' shall be used). The TSS must include any error states that they TSF may enter when self-tests fail, and the conditions and actions necessary to exit the error states and resume normal operation. The evaluator shall verify that the TSS indicates these self-tests are run at start-up automatically, and do not involve any inputs from or actions by the user or operator.

The evaluator shall inspect the list of self-tests in the TSS and verify that it includes algorithm self-tests. The algorithm self-tests will typically be conducted using known answer tests.

Section 6.6 of the ST states the TOE performs power-on self-tests including known answer tests. Table 36 in the TSS explains the power-on self-tests and the crypto libraries that perform the tests.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.17 TSF Self-Test (VPNC24:FPT_TST_EXT.1/VPN)

2.6.17.1 VPNC24:FPT_TST_EXT.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.17.2 VPNC24:FPT_TST_EXT.1.2/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Except for where it is explicitly noted, the evaluator is expected to check the following information regardless of whether the functionality is implemented by the TOE or by the TOE platform.

The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF on start-up; this description should include an outline of what the tests are actually doing (e.g., rather than saying 'memory is tested', a description similar to 'memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written' shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly. If some of the tests are performed by the TOE platform, the evaluator shall check the TSS to ensure that those tests are identified, and that the ST for each platform contains a description of those tests. Note that the tests that are required by this component are those that support security functionality in the VPN Client PP-Module, which may not correspond to the set of all self-tests contained in the platform STs.

The evaluator shall examine the TSS to ensure that it describes how the integrity of stored TSF executable code is cryptographically verified when it is loaded for execution. The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the integrity of stored TSF executable code has not been compromised. The evaluator shall check to ensure that the cryptographic requirements listed are consistent with the description of the integrity verification process.

The evaluator also ensures that the TSS (or the operational guidance) describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases. For checks implemented entirely by the platform, the evaluator ensures that the operational guidance for the TOE references or includes the platform-specific guidance for each platform listed in the ST.

Section 6.6 of the ST identifies that the TOE verifies the integrity of its executable code (libcharon.so) upon load and prior to execution (the TOE Platform loads the executable whenever a VPN connection is requested/attempted). It also references the previously described self-tests.

Component Guidance Assurance Activities: Except for where it is explicitly noted, the evaluator is expected to check the following information regardless of whether the functionality is implemented by the TOE or by the TOE platform.

If not present in the TSS, the evaluator ensures that the operational guidance describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases. For checks implemented entirely by the platform, the evaluator ensures that the operational guidance for the TOE references or includes the platform-specific guidance for each platform listed in the ST.

Section 8.1 of the Admin Guide explains errors a user might encounter. It instructs the user to seek help from the Enterprise Management or if necessary, return the device to the supplier.

Component Testing Assurance Activities: Except for where it is explicitly noted, the evaluator is expected to check the following information regardless of whether the functionality is implemented by the TOE or by the TOE platform.

The evaluator shall perform the following tests:

Test 1: The evaluator performs the integrity check on a known good TSF executable and verifies that the check is successful.

Test 2: The evaluator modifies the TSF executable, performs the integrity check on the modified TSF executable and verifies that the check fails.

Test 1 – See WLANC10:FPT_TST_EXT.1 test 2 which demonstrates the TOE boots properly with no integrity errors and fails to boot when integrity errors are detected.

Test 2 – See WLANC10:FPT_TST_EXT.1 test 2 which demonstrates the TOE boots properly with no integrity errors and fails to boot when integrity errors are detected.

2.6.18 TSF Integrity Checking (Post-Kernel) (MDFPP33:FPT_TST_EXT.2/POSTKERNEL)

2.6.18.1 MDFPP33:FPT_TST_EXT.2.1/POSTKERNEL

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluation activity shall be completed in conjunction with FPT_TST_EXT.2/PREKERNEL for all executable code specified.

See MDFPP33:FPT_TST_EXT.2/PREKERNEL

2.6.19 TSF Integrity Checking (Pre-Kernel) (MDFPP33:FPT_TST_EXT.2/PREKERNEL)

2.6.19.1 MDFPP33:FPT_TST_EXT.2.1/PREKERNEL

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS section of the ST includes a description of the boot procedures, including a description of the entire bootchain, of the software for the TSF's Application Processor. The evaluator shall ensure that before loading the bootloader(s) for the operating system and the kernel, all bootloaders and the kernel software itself is cryptographically verified. For each additional category of executable code verified before execution, the evaluator shall verify that the description in the TSS describes how that software is cryptographically verified.

The evaluator shall verify that the TSS contains a justification for the protection of the cryptographic key or hash, preventing it from being modified by unverified or unauthenticated software. The evaluator shall verify that the TSS contains a description of the protection afforded to the mechanism performing the cryptographic verification.

The evaluator shall verify that the TSS describes each auxiliary boot mode available on the TOE during the boot procedures. The evaluator shall verify that, for each auxiliary boot mode, a description of the cryptographic integrity of the executed code through the kernel is verified before each execution.

Section 6.6 of the ST explains the TOE ensures a secure boot process in which the TOE verifies the digital signature of the bootloader software for the Application Processor (using a public key whose hash resides in the processor's internal fuses) before transferring control. The bootloader, in turn, verifies the signature of the Linux kernel (either the primary or the recovery kernel) it loads.

Once the kernel has been loaded, the TOE uses dm-verity in EIO mode to protect the integrity of the system partition. After verifying the digital signature of the dm-verity hash tree (using the same public key that verifies the kernel image), the TOE will reject the loading of file system blocks where the integrity does not match, and return an I/O error (as if the block were unreadable).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation Activity Note: The following tests require the vendor to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

The evaluator shall perform the following tests:

Test 99: The evaluator shall perform actions to cause TSF software to load and observe that the integrity mechanism does not flag any executables as containing integrity errors and that the TOE properly boots.

Test 100: The evaluator shall modify a TSF executable that is integrity protected and cause that executable to be successfully loaded by the TSF. The evaluator observes that an integrity violation is triggered and the TOE does not boot. (Care must be taken so that the integrity violation is determined to be the cause of the failure to load the module, and not the fact that the module was modified so that it was rendered unable to run because its format was corrupt).

Test 101: [conditional] If the ST author indicates that the integrity verification is performed using a public key, the evaluator shall verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1. The evaluator shall digitally sign the TSF executable with a certificate that does not have the Code Signing purpose in the extendedKeyUsage field and verify that an integrity violation is triggered. The evaluator shall repeat the test using a certificate that contains the Code Signing purpose and verify that the integrity verification succeeds. Ideally, the two certificates should be identical except for the extendedKeyUsage field.

Tests 99 & 100– See WLANC10:FPT_TST_EXT.3 test 2 which demonstrates the TOE boots properly with no integrity errors and fails to boot when integrity errors are detected.

Test 101 – Not applicable

2.6.20 TSF Cryptographic Functionality Testing (WLAN Client) (WLANC10:FPT_TST_EXT.3/WLAN)

2.6.20.1 WLANC10:FPT_TST_EXT.3.1/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.20.2 WLANC10:FPT_TST_EXT.3.2/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it details the self tests that are run by the TSF on start-up; this description should include an outline of what the tests are actually doing (e.g., rather than saying 'memory is tested', a description similar to 'memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written' shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

The evaluator shall examine the TSS to ensure that it describes how to verify the integrity of stored TSF executable code when it is loaded for execution. The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the integrity of stored TSF executable code has not been compromised. The evaluator also ensures that the TSS (or the operational guidance) describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases.

Section 6.6 of the ST explains the TOE platform performs the self-tests mentioned in MDFPP33:FPT_TST_EXT.1 to ensure the integrity of the WLAN client (wpa_supplicant).

Component Guidance Assurance Activities: The evaluator shall ensure that the operational guidance describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases.

Section 8.1 of the Admin Guide explains errors a user might encounter. It instructs the user to seek help from the Enterprise Management or if necessary return the device to the supplier.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall perform the integrity check on a known good TSF executable and verify that the check is successful.

Test 2: The evaluator shall modify the TSF executable, perform the integrity check on the modified TSF executable, and verify that the check fails.

Test 1 – See test 2 where a corrupted binary is used to ensure the corruption is detected and then the good/correct binary is restored to show that the TOE is working properly (after having passed the integrity test).

Test 2 – The evaluator modified a system file identified in the ST. After the file was modified, the evaluator rebooted the device. The evaluator found that in each case the TOE devices would not boot until after the original binaries were restored and the devices reset.

2.6.21 TSF Version Query (MDFPP33:FPT_TUD_EXT.1)

2.6.21.1 MDFPP33:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.21.2 MDFPP33:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.21.3 MDFPP33:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall establish a test environment consisting of the Mobile Device and any supporting software that demonstrates usage of the management functions. This can be test software from the developer, a reference implementation of management software from the developer, or other commercially available software. The evaluator shall set up the Mobile Device and the other software to exercise the management functions according to the provided guidance documentation.

Test 102: Using the AGD guidance provided, the evaluator shall test that the administrator and user can query:

- The current version of the TSF operating system and any firmware that can be updated separately

- The hardware model of the TSF

- The current version of all installed mobile applications

The evaluator must review manufacturer documentation to ensure that the hardware model identifier is sufficient to identify the hardware which comprises the device.

Test 1 – Section 1.4 of the Admin Guide identifies the evaluated devices. The software versions are identified as the following for the devices that underwent testing:

Device Name	Chipset Vendor	SoC	Arch	Kernel Version	Build Number
Galaxy Z Flip6 5G	Qualcomm	Snapdragon 8 Gen 3 Mobile Platform	ARMv8	6.1	UP1A.231005.007
Galaxy A52 5G	Samsung	Snapdragon 750	ARMv8	4.19	UP1A.231005.007
Galaxy A53 5G	Samsung	Exynos 1280	ARMv8	5.15	UP1A.231005.007
Galaxy S24 FE	Samsung	Exynos 2400	ARMv8	6.1	UP1A.231005.007

This version information was verified in the UI. Application versions were determined using the Application Manager interface. This test was repeated as an administrator.

The administrator test used the MDM tool to query the model and version, list of installed apps, and specific app info (including version).

2.6.22 TSF Update Verification (MDFPP33:FPT_TUD_EXT.2)

2.6.22.1 MDFPP33:FPT_TUD_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.22.2 MDFPP33:FPT_TUD_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.22.3 MDFPP33:FPT_TUD_EXT.2.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS section of the ST describes all TSF software update mechanisms for updating the system software. The evaluator shall verify that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails. The evaluator shall verify that all software and firmware involved in updating the TSF is described and, if multiple stages and software are indicated, that the software/firmware responsible for each stage is indicated and that the stages which perform signature verification of the update are identified.

The evaluator shall verify that the TSS describes the method by which the digital signature is verified and that the public key used to verify the signature is either hardware-protected or is validated to chain to a public key in the Trust Anchor Database. If hardware-protection is selected, the evaluator shall verify that the method of hardware-protection is described and that the ST author has justified why the public key may not be modified by unauthorized parties.

[conditional] If the ST author indicates that software updates to system software running on other processors is verified, the evaluator shall verify that these other processors are listed in the TSS and that the description includes the software update mechanism for these processors, if different than the update mechanism for the software executing on the Application Processor.

[conditional] If the ST author indicates that the public key is used for software update digital signature verification, the evaluator shall verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1 and a check for the Code Signing purpose in the extendedKeyUsage.

Section 6.6 of the ST states when in CC mode, the TOE verifies all updates to the TOE software using a public key (FOTA public key) chaining ultimately to the Secure Boot Public Key (SBPK), a hardware protected key whose SHA-256 hash resides inside the application processor (note that when not in CC mode, the TOE allows updates to the TOE software through the Download mode of the bootloader). After verifying an update’s FOTA signature, the TOE will then install those updates to the TOE. The TOE will check a new image to ensure that the image is not older than the current image, and if so, the TOE will reject the new image and not update the TOE software.

The application processing verifies the bootloader’s authenticity and integrity (thus tying the bootloader and subsequent stages to a hardware root of trust: the SHA-256 hash of the SBPK, which cannot be reprogrammed after the “write-enable” fuse has been blown).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall verify that the developer has provided evidence that the following tests were performed for each available update mechanism:

Test 103: The tester shall try to install an update without the digital signature and shall verify that installation fails. The tester shall attempt to install an update with digital signature, and verify that installation succeeds.

Test 104: The tester shall digitally sign the update with a key disallowed by the device and verify that installation fails. The tester shall attempt to install an update signed with the allowed key and verify that installation succeeds.

Test 105: [conditional] The tester shall digitally sign the update with an invalid certificate and verify that update installation fails. The tester attempt to install an update that was digitally signed using a valid certificate and a certificate that contains the purpose and verify that the update installation succeeds.

Test 106: [conditional] The tester shall repeat these tests for the software executing on each processor listed in the first selection. The tester shall attempt to install an update without the digital signature and shall verify that installation fails. The tester shall attempt to install an update with digital signature, and verify that installation succeeds.

Test 103 – The developer provided three updates – one signed, one unsigned, and one signed by a disallowed key. The evaluator loaded the updates on the device and attempted to install each device. The evaluator attempted to install the unsigned update and failed with a signature verification error. The evaluator then attempted to install the disallowed key update and failed. The evaluator attempted to install the signed update and succeeded.

Test 104 – This was tested as part of test 1.

Test 105 – This test is not applicable.

Test 106 – See test case 1. The developer provided updates, including updated files for each processor: AP, Bootloader, Communication Processor, and Carrier Specific Configuration. The integrity check is on the entire update package and not on specific files in the package.

2.6.23 Application Signing (MDFPP33:FPT_TUD_EXT.3)

2.6.23.1 MDFPP33:FPT_TUD_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes how mobile application software is verified at installation. The evaluator shall ensure that this method uses a digital signature.

Section 6.6 of the ST states The TOE requires that all applications be signed before they can be installed. Android uses a file format called APK to package the application installation. The contents of the APK are hashed and then signed, with the resulting APK Signing Block then inserted into the APK or saved as a separate file (depending on the APK Signature Scheme in use).

Once an application has been downloaded, the Signing Block information is used to verify the software before installation. The app contents are verified against the hash values that have been calculated and signed before approving the installation.

In addition to the hash check, there are several rules related to the contents of the Signing Block which can be found here: https://source.android.com/security/apksigning/v3.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation Activity Note: The following test does not have to be tested using the commercial application store.

Test 107: The evaluator shall write, or the developer shall provide access to, an application. The evaluator shall try to install this application without a digitally signature and shall verify that installation fails. The evaluator shall attempt to install a digitally signed application, and verify that installation succeeds.

Test 107 – The developer provided the evaluator 2 applications – one signed and one not signed. The evaluator attempted to install the signed application and was successful. The evaluator attempted to install the unsigned application and it failed with a "no certificate" message.

2.6.24 Trusted Update Verification (MDFPP33:FPT_TUD_EXT.6)

2.6.24.1 MDFPP33:FPT_TUD_EXT.6.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the mechanism that prevents the TSF from installing software updates that are an older version that the currently installed version.

Section 6.6 of the ST states the TOE maintains a monotonic version counter for the TOE software. Before a new update can be installed, the version of the new software is verified to the version counter. If the new image is older than the current version, the update will be rejected and not installed; only version numbers that are equal to or greater than the currently set number can be installed on the device.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall repeat the following tests to cover all allowed software update mechanisms as described in the TSS. For example, if the update mechanism replaces an entire partition containing many separate code files, the evaluator does not need to repeat the test for each individual file.

Test 141: The evaluator shall attempt to install an earlier version of software (as determined by the manufacturer). The evaluator shall verify that this attempt fails by checking the version identifiers or cryptographic hashes of the privileged software against those previously recorded and checking that the values have not changed.

Test 142: The evaluator shall attempt to install a current or later version and shall verify that the update succeeds.

Test 141 – The evaluator attempted to install a previous software version on each device. In each case, the install produced an error and was denied.

Test 142 - The evaluator attempted to install a current software version on each device. The install was successful.

2.7 TOE ACCESS (FTA)

2.7.1 TSF- AND USER-INITIATED LOCKED STATE (MDFPP33:FTA_SSL_EXT.1)

2.7.1.1 MDFPP33:FTA_SSL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.2 MDFPP33:FTA_SSL_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.3 MDFPP33:FTA_SSL_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify the TSS describes the actions performed upon transitioning to the locked state.

Section 6.7 of the ST states the TOE transitions to its locked state either immediately after a user initiates a lock by pressing the power button or after a configurable period of inactivity. As part of that transition, the TOE will display a lock screen to obscure the previous contents; however, the TOE's lock screen still allows a user to perform the functions listed in section 5.1.4.18 of the ST before authenticating. However, without authenticating first, a user cannot perform any related actions based upon these notifications (for example, they cannot respond to emails, calendar appointment requests, or text messages) other than answering an incoming phone call.

On power up the TOE boots to the device lock screen. On the first boot, the user can only make emergency calls, receive calls, enter their password or see notifications from apps that do not require user authentication (apps that have requested the use of Device Encrypted storage during installation).

> KNOX - Section 6.9 of the ST explains the work profile transitions to its locked state either immediately after a user initiates a lock by pressing the work profile lock button from the notification bar or after a configurable period of inactivity, and as part of that transition, the work profile will display a lock screen to obscure the previous contents. When the work profile is locked, it can still display calendar appointments and other notifications allowed by the administrator to be shown outside the work profile (in the notification area). However, without authenticating first to the work profile, a user cannot perform any related actions based upon these work profile notifications (they cannot respond to emails, calendar appointments, or text messages).

The work profile timeout is independent of the TOE timeout and as such can be set to different values.

Component Guidance Assurance Activities: The evaluation shall verify that the AGD guidance describes the method of setting the inactivity interval and of commanding a lock. The evaluator shall verify that the TSS describes the information allowed to be displayed to unauthorized users.

Section 4.1 of the Admin Guide provides a discussion on interfaces for locking the device as well as the KNOX interfaces associated with locking.

Component Testing Assurance Activities: Test 108: The evaluator shall configure the TSF to transition to the locked state after a time of inactivity (FMT_SMF.1) according to the AGD guidance. The evaluator shall wait until the TSF locks and verify that the display is cleared or overwritten and that the only actions allowed in the locked state are unlocking the session and those actions specified in FIA_UAU_EXT.2.

Test 109: The evaluator shall command the TSF to transition to the locked state according to the AGD guidance as both the user and the administrator. The evaluator shall wait until the TSF locks and verify that the display is cleared or overwritten and that the only actions allowed in the locked state are unlocking the session and those actions specified in FIA_UAU_EXT.2.

Test 108 – The TOE was alternately configured with 5 and 10 minute session timeout settings and after showing the configured time, the TOE was left inactive for the configured period of time to demonstrate that it locked at that time as expected. The test was repeated for a Knox container.

The lockscreen login display shows a number of device status indicators and allows the following list of functions to be performed:

- Take screen shots (stored internally)
- Enter password to unlock

- Make/receive emergency calls
- Take pictures (stored internally) – unless the camera was disabled
- Turn the TOE off
- Restart the TOE
- Enable emergency mode (which saves power)
- See notifications (note that some notifications identify actions, for example to view a screenshot; however, selecting those notifications highlights the password prompt and require the password to access that data)
- Configure sound, vibrate, or mute
- Set the volume (up and down) for various sound categories
- List and attempt to start configured Edge apps (on the Edge devices) – however, requires authentication to actually use the apps
- Access notification widgets (without authentication):
  - Always on Display toggle
  - Flashlight toggle
  - Do not disturb toggle
  - Auto rotate toggle
  - Sound (on, mute, vibrate)
  - Edge lighting toggle

Test 109 – The evaluator configured the TOE to lock immediately when the display is turned off. The evaluator turned off the display (pressed the power button briefly) to demonstrate that it locked immediately as expected. The test was repeated for a Knox container.

2.7.2 Default TOE Access Banners (MDFPP33:FTA_TAB.1)

2.7.2.1 MDFPP33:FTA_TAB.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The TSS shall describe when the banner is displayed.

Section 6.7 of the ST states the TOE can be configured by an administrator to display a message on the lock screen using an MDM.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall also perform the following test:

Test 110: The evaluator follows the operational guidance to configure a notice and consent warning message. The evaluator shall then start up or unlock the TSF. The evaluator shall verify that the notice and consent warning message is displayed in each instance described in the TSS.

Test 110 - The evaluator used the MDM application to configure a banner message. The evaluator then pressed the power button to lock the TOE and then pressed it again to initiate an unlock action. The banner was found on the display. The evaluator also configured a custom message and demonstrated it was displayed as a banner.

2.7.3 Wireless Network Access (WLANC10:FTA_WSE_EXT.1)

2.7.3.1 WLANC10:FTA_WSE_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it defines SSIDs as the attribute to specify acceptable networks.

Section 6.7 of the ST states the TOE allows an administrator to specify (using an MDM) a list of wireless networks (SSIDs) to which the user may direct the TOE to connect. When not enrolled with an MDM, the TOE allows the user to control to which wireless networks the TOE should connect, but does not provide an explicit list of such networks, rather the user may scan for available wireless network (or directly enter a specific wireless network), and then connect. Once a user has connected to a wireless network, the TOE will automatically reconnect to that network when in range and the user has enabled the TOE’s Wi-Fi radio.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it contains guidance for configuring the list of SSIDs that the WLAN Client is able to connect to.

The embedded spreadsheet in Section 3.3 of the Admin Guide explains how to configure wireless networks.

Component Testing Assurance Activities: The evaluator shall perform the following tests for each attribute:

Test 1: The evaluator configures the TOE to allow a connection to a wireless network with a specific SSID. The evaluator configures the test environment such that the allowed SSID and an SSID that is not allowed are both 'visible' to the TOE. The evaluator shall demonstrate that they can successfully establish a connection with the allowed SSID. The evaluator shall then attempt to establish a session with the disallowed SSID and observe that the attempt fails.

Test 1 – See FMT_MOF_EXT.1 test 70, function 49 where various authorized and unauthorized cases are tested including blocked, black and white listing

2.8 Trusted path/channels (FTP)

2.8.1 Bluetooth Encryption (BT10:FTP_BLT_EXT.1)

2.8.1.1 BT10:FTP_BLT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.2 BT10:FTP_BLT_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the use of encryption, the specific Bluetooth protocol(s) it applies to, and whether it is enabled by default.

The evaluator shall verify that the TSS includes the protocol used for encryption of the transmitted data and the key generation mechanism used.

Section 6.8 of the ST states the TOE provides support for both Bluetooth BR/EDR and Bluetooth LE connections. The TSF ensures that all connections are encrypted by default using keys of at least 128-bits (regardless of the type of connection).

Component Guidance Assurance Activities: The evaluator shall verify that the operational guidance includes instructions on how to configure the TOE to require the use of encryption during data transmission (unless this behavior is enforced by default).

Bluetooth encryption is enabled by default. No configuration is required by the administrator.

Component Testing Assurance Activities: There are no test EAs for this component. Testing for this SFR is addressed through the evaluation of FTP_BLT_EXT.3/BR and, if claimed, FTP_BLT_EXT.3/LE.

See FTP_BLT_EXT.3/BR and FTP_BLT_EXT.3/LE.

2.8.2 Persistence of Bluetooth Encryption (BT10:FTP_BLT_EXT.2)

2.8.2.1 BT10:FTP_BLT_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the TSF's behavior if a remote device stops encryption while connected to the TOE.

Section 6.8 of the ST explains that since the TOE requires an encrypted connection between itself and another Bluetooth device, if the remote device stops encryption, the TSF will force the termination of the connection. A new connection should re-establish the encrypted channel (and if not, the connection will not be successful).

Component Guidance Assurance Activities: The evaluator shall verify that the operational guidance describes how to enable/disable encryption (if configurable).

Bluetooth encryption is enabled by default. No configuration is required by the administrator.

Component Testing Assurance Activities: Test 1: The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:

Step 1: Initiate pairing with the TOE from a remote Bluetooth device that has been configured to have a minimum encryption key size that is equal to or greater than that of the TOE.

Step 2: After pairing has successfully finished and while a connection exists between the TOE and the remote device, turn off encryption on the remote device. This can be done using commercially-available tools.

Step 3: Verify that the TOE either restarts encryption with the remote device or terminates the connection with the remote device.

The evaluator established a Bluetooth connection between two devices and observed that encryption was enabled in the packet capture. The evaluator then disabled encryption on one device and demonstrated via a packet capture that the TOE terminated the connection after receiving the encryption off command.

2.8.3 Bluetooth Encryption Parameters (BR/EDR) - per TD0640 (BT10:FTP_BLT_EXT.3/BR)

2.8.3.1 BT10:FTP_BLT_EXT.3.1/BR

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS and verify that it specifies the minimum key size for BR/EDR encryption, whether this value is configurable, and the mechanism by which the TOE will not negotiate keys sizes smaller than the minimum.

Section 6.8 of the ST explains the TOE provides support for both Bluetooth BR/EDR and Bluetooth LE connections. The TSF ensures that all connections are encrypted by default using keys of at least 128-bits (regardless of the type of connection).

Component Guidance Assurance Activities: The evaluator shall verify that the guidance includes instructions on how to configure the minimum encryption key size for BR/EDR encryption, if configurable.

Bluetooth encryption is enabled by default including the key size. No configuration is required by the administrator.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:

Step 1: Initiate BR/EDR pairing with the TOE from a remote Bluetooth device that has been configured to have a minimum encryption key size that is equal to or greater than that of the TOE. This can be done using certain commercially-available tools that can send the appropriate command to certain commercially-available Bluetooth controllers.

Step 2: Use a Bluetooth packet sniffer to verify that the encryption key size negotiated for the connection is at least as large as the minimum encryption key size defined for the TOE.

Test 2: (conditional): If the encryption key size is configurable, configure the TOE to support a different minimum key size, then repeat Test 1 and verify that the negotiated key size is at least as large as the new minimum value.

Test 3: The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:

Step 1: Initiate BR/EDR pairing with the TOE from a remote Bluetooth device that has been configured to have a maximum encryption key size of 1 byte. This can be done using certain commercially-available tools that can send the appropriate command to certain commercially-available Bluetooth controllers.

Step 2: Verify that the encryption key size suggested by the remote device is not accepted by the TOE and that the connection is not completed.

Test 1 – The evaluator established a BT/BR connection with another device. The evaluator was able to identify the negotiated Linkkey was 16 bytes in length as expected.

Test 2 - Not applicable - the TOE does not support changing the Bluetooth/BR key size.

Test 3 – The developer provided a test device that always requires a 1-byte link key. The TOE failed on the attempt to connect with a 1-byte key.

2.8.4 Bluetooth Encryption Parameters (LE) (BT10:FTP_BLT_EXT.3/LE)

2.8.4.1 BT10:FTP_BLT_EXT.3.1/LE

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS and verify that it specifies the minimum key size for LE encryption, whether this value is configurable, and the mechanism by which the TOE will not negotiate keys sizes smaller than the minimum.

Section 6.8 of the ST explains the TOE provides support for both Bluetooth BR/EDR and Bluetooth LE connections. The TSF ensures that all connections are encrypted by default using keys of at least 128-bits (regardless of the type of connection).

Component Guidance Assurance Activities: The evaluator shall verify that the guidance includes instructions on how to configure the minimum encryption key size for LE encryption, if configurable.

Bluetooth encryption is enabled by default including the key size. No configuration is required by the administrator.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:

Step 1: Initiate LE pairing with the TOE from a remote Bluetooth device that has been configured to have a minimum encryption key size that is equal to or greater than that of the TOE. This can be done using certain commercially-available tools that can send the appropriate command to certain commercially-available Bluetooth controllers.

Step 2: Use a Bluetooth packet sniffer to verify that the encryption key size negotiated for the connection is at least as large as the minimum encryption key size defined for the TOE.

Test 2: (conditional): If the encryption key size is configurable, configure the TOE to support a different minimum key size, then repeat Test 1 and verify that the negotiated key size is at least as large as the new minimum value.

Test 3: The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:

Step 1: Initiate LE pairing with the TOE from a remote Bluetooth device that has been configured to have a maximum encryption key size of 1 byte. This can be done using certain commercially-available tools that can send the appropriate command to certain commercially-available Bluetooth controllers.

Step 2: Verify that the encryption key size suggested by the remote device is not accepted by the TOE and that the connection is not completed.

Test 1 - The evaluator created an application to establish a BT/LE connection. The evaluator was able to identify the negotiated Linkkey was 16 bytes in length as expected.

Test 2 - Not applicable - the TOE does not support changing the Bluetooth/LE key size.

Test 3 - The developer provided a test device that always requires a 1-byte link key and advertises Bluetooth/LE. The TOE failed on the attempt to connect with a 1-byte key.

2.8.5 Trusted Channel Communication (MDFPP33:FTP_ITC_EXT.1)

2.8.5.1 MDFPP33:FTP_ITC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.5.2 MDFPP33:FTP_ITC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.5.3 MDFPP33:FTP_ITC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to access points, VPN Gateways, and other trusted IT products in terms of the cryptographic protocols specified in the requirement, along with TOE-specific options or procedures that might not be reflected in the specifications. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST.

If OTA updates are selected, the TSS shall describe which trusted channel protocol is initiated by the TOE and is used for updates.

Section 6.8 of the ST states the TOE provides secured (encrypted and mutually authenticated) communication channels between itself and other trusted IT products through the use of 802.11-2012, 802.1X, EAP-TLS, TLS, HTTPS, and IPsec. The ST has requirements for 802.11-2012, 802.1X, EAP-TLS, TLS, HTTPS, and IPsec (MOD_VPN_CLI_V2.4:FCS_CKM.1, FCS_CKM.2.1, FIA_PAE_EXT.1.1, FCS_HTTPS_EXT.1, FCS_TLSC_EXT.1, WLANC10:FCS_TLSC_EXT.1, and MOD_VPN_CLI_V2.4: FCS_IPSEC_EXT.1). The cryptographic protocols are described in each section of the TSS that addresses their requirement. There are no TOE-specific options described. The TOE is also being validated against the PP-Module for Virtual Private Network (VPN) Clients.

In Section 4.2 of the Admin Guide, the user is instructed that detailed instructions for connecting to Wi-Fi networks can be found under the “Connections” section of the guide for the specific device. Additionally, guidance is provided to address what the user should do if connection is unintentionally dropped (re-establish the connection).

Component Guidance Assurance Activities: The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to access points, VPN Gateways, and other trusted IT products.

The Admin Guide contains instructions for making Wi-Fi connections in Section 4.2 and VPN Gateways in Section 4.6.

Component Testing Assurance Activities: The evaluator shall also perform the following tests for each protocol listed:

Test 111: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data are not sent in plaintext and that a protocol analyzer identifies the traffic as the protocol under testing.

Test 112: [conditional] If IPsec is selected, the evaluator shall ensure that the TOE is able to initiate communications with a VPN Gateway, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 113: [conditional] If OTA updates are selected, the evaluator shall trigger an update request according to the operational guidance and shall ensure that the communication is successful.

Test 114: For any other selected protocol (not tested in Test 1, 2, or 3), the evaluator shall ensure that the TOE is able to initiate communications with a trusted IT product using the protocol, setting up the connection as described in the operational guidance and ensuring that the communication is successful.

Test 111 -This has been tested in other test cases:

- For 802.11-2012 encryption, see Test Case WLANC10:FCS_CKM.1/WPA test case 2 where Wi-Fi encryption is demonstrated
- For 802.1X and EAP-TLS, see Test Case WLANC10:FCS_TLSC_EXT.1/WLAN test case 1 where this encryption is demonstrated
- For mutually-authenticated TLS, see Test Case FCS_TLSC_EXT.2.1 test case 1
- For IPsec, see FCS_IPSEC_EXT.1.1 test case 1
- For HTTPS, see FCS_TLSC_EXT.1.1 test case 1

Test 112 – See Test Case VPNC24:FCS_IPSEC_EXT.1.1 for IPsec where all the connections are initiated from the TOE. See also Test Case VPNC24:FDP_IFC_EXT.1 test case 1 where a VPN connection is established and traffic is sent (ESP packets in the packet capture) for all TOE devices.

Test 113 – Not applicable. OTA updates are not claimed.

Test 114 – Not applicable since all the secure channels are otherwise claimed and tested

2.8.6 Trusted Channel Communication (VPNC24:FTP_ITC_EXT.1)

2.8.6.1 VPNC24:FTP_ITC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.6.2 VPNC24:FTP_ITC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.6.3 VPNC24:FTP_ITC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to access points, VPN Gateways, and other trusted IT products in terms of the cryptographic protocols specified in the requirement, along with TOE-specific options or procedures that might not be reflected in the specifications. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST. The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to access points, VPN Gateways, and other trusted IT products.

If OTA updates are selected, the TSS shall describe which trusted channel protocol is initiated by the TOE and is used for updates.

See MDFPP33:FTP_ITC_EXT.1

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Refer to the EAs for FTP_ITC_EXT.1 in the MDF PP (included below). The only change to this SFR is that some selections are mandated, therefore the corresponding testing is mandatory. The actual testing for those selections is not changed.

The evaluator shall also perform the following tests for each protocol listed:

Test 1: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data are not sent in plaintext and that a protocol analyzer identifies the traffic as the protocol under testing.

Test 2: [conditional] If IPsec is selected (and the TSF includes a native VPN client), the evaluator shall ensure that the TOE is able to initiate communications with a VPN Gateway, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Test 3: [conditional] If OTA updates are selected, the evaluator shall trigger an update request according to the operational guidance and shall ensure that the communication is successful.

Test 4: For any other selected protocol (not tested in Test 1, 2, or 3), the evaluator shall ensure that the TOE is able to initiate communications with a trusted IT product using the protocol, setting up the connection as described in the operational guidance and ensuring that the communication is successful.

See MDFPP33:FTP_ITC_EXT.1

2.8.7 Trusted Channel Communication (Wireless LAN) (WLANC10:FTP_ITC.1/WLAN)

2.8.7.1 WLANC10:FTP_ITC.1.1/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.7.2 WLANC10:FTP_ITC.1.2/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.7.3 WLANC10:FTP_ITC.1.3/WLAN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to an access point in terms of the cryptographic protocols specified in the requirement, along with TOE-specific options or procedures that might not be reflected in the specification. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST.

This was addressed in the description for MDFPP33:FTP_ITC_EXT.1.

Component Guidance Assurance Activities: The evaluator shall confirm that the operational guidance includes instructions for establishing the connection to the access point and that it includes recovery instructions should a connection be unintentionally broken.

In Section 4.2 of the Admin Guide, the user is instructed that detailed instructions for connecting to Wi-Fi networks can be found under the "Connections" section of the guide for the specific device. Additionally, guidance is provided to address what the user should do if connection is unintentionally dropped (re-establish the connection).

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall ensure that the TOE is able to initiate communications with an access point using the protocols specified in the requirement by setting up the connections as described in the operational guidance and ensuring that communications are successful.

Test 2: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, modification of the channel data is detected by the TOE.

Test 4: The evaluators shall physically interrupt the connection from the TOE to the access point (e.g., moving the TOE host out of range of the access point, turning the access point off). The evaluators shall ensure that subsequent communications are appropriately protected, at a minimum in the case of any attempts to automatically resume the connection or connect to a new access point.

Further evaluation activities are associated with the specific protocols.

Test 1 – See Test Case WLANC10:FTP_ITC_EXT.1 test case 4 where Access Point connections are made with a pre-shared key. See also WLANC10:FCS_TLSC_EXT.1 test case 1 where several Access Point connections are made using EAP-TLS.

Test 2 – See WLANC10:FCS_CKM.1 test case 2 where it can be seen that packets are protected using 802.11 once a connection is made with EAPOL based on a pre-shared key. Additionally, a wireless packet capture was provided showing the 802.11 packets in the air are encrypted once a connection is made with EAPOL based on EAP-TLS.

Test 3 – The evaluator modified traffic from the access point and observed in the packet capture that the traffic was dropped.

Test 4 – The evaluator connected to an access point and then interrupted power to the access point. When the power was restored to the access point, the TOE and access point re-created a secure channel.

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the claimed Protection Profile that correspond with Security Assurance Requirements.

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

Assurance Activities: There are no specific evaluation activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the evaluation activities described in Section 5.1 Security Functional Requirements, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other evaluation activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

Assurance Activities: Some of the contents of the operational guidance are verified by the evaluation activities in Section 5.1 Security Functional Requirements and evaluation of the TOE according to the [CEM]. The following additional information is also required.

The operational guidance shall contain a list of natively installed applications and any relevant version numbers. If any third party vendors are permitted to install applications before purchase by the end user or enterprise, these applications shall also be listed.The operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

The documentation must describe the process for verifying updates to the TOE by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

- Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

- Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature.

The TOE will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

Section 3.1 of the Admin Guide explains Samsung has chosen to utilize NIST-validated cryptographic algorithms within the cryptographic modules on its devices for the Common Criteria configuration. These algorithms are made available for use by applications installed on the device through the normal Android Framework APIs. The APIs which provide access to FIPS-validated algorithms are detailed in Section 6. The specific FIPS libraries are identified in the Admin Guide.

Section 7.2 of the Admin Guide discusses how to update the device. It explains how to get an update and how updates are signed.

Section 2.2 of the Admin Guide provides a list of the evaluated capabilities of the device.

3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activities: As indicated in the introduction above, there are significant expectations with respect to the documentationâ€”especially when configuring the operational environment to support TOE functional requirements. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms claimed for the TOE in the ST.

Section 3.3.1 of the Admin Guide explains how to put the device into CC mode.

3.3 Life-cycle support (ALC)

3.3.1 Labeling of the TOE (ALC_CMC.1)

Assurance Activities: The evaluator shall check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

The evaluator verified that the ST, TOE and Guidance are all labeled with the same versions. The evaluator checked the TOE version during testing by examining the actual devices used for testing.

3.3.2 TOE CM Coverage (ALC_CMS.1)

Assurance Activities: The evaluator shall ensure that the developer has identified (in public-facing development guidance for their platform) one or more development environments appropriate for use in developing applications for the developer's platform. For each of these development environments, the developer shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environments are invoked (e.g., compiler and linker flags). The evaluator shall ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled.

The evaluator shall ensure that the TSF is uniquely identified (with respect to other products from the TSF vendor), and that documentation provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

See section 3.3.1 above for an explanation of how all CM items are identified. In regards to development environments, developers who wish to develop apps for the devices can go to http://developer.android.com/index.html and specially https://developer.android.com/privacy-and-security/security-tips for security development practices as referenced in the Admin Guide section 6.7 .

3.3.3 Timely Security Updates (ALC_TSU_EXT.1)

Assurance Activities: The evaluator shall verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates. The evaluator shall verify that this description addresses the TOE OS, the firmware, and bundled applications, each. The evaluator shall also verify that, in addition to the TOE developer's process, any carrier or other third-party processes are also addressed in the description. The evaluator shall also verify that each mechanism for deployment of security updates is described.

The evaluator shall verify that, for each deployment mechanism described for the update process, the TSS lists a time between public disclosure of a vulnerability and public availability of the security update to the TOE patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator shall verify that this time is expressed in a number or range of days.

The evaluator shall verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the TOE. The evaluator shall verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.

The evaluator shall verify that the description includes where users can seek information about the availability of new security updates including details of the specific public vulnerabilities corrected by each update. The evaluator shall verify that the description includes the minimum amount of time that the TOE is expected to be supported with security updates, and the process by which users can seek information about when the TOE is no longer expected to receive security updates.

Section 6.6 of the ST provides a discussion about timely updates. Samsung utilizes industry best practices to ensure their devices are patched to mitigate security flaws. Samsung provides a web portal for reporting potential security issues (https://security.samsungmobile.com/securityReporting.smsb) with instructions about how to securely contact and communicate with Samsung. As an Android OEM, Samsung also works with Google on reported Android issues (https://source.android.com/source/report-bugs.html) to ensure customer devices are secure.

Samsung will create updates and patches to resolve reported issues as quickly as possible, at which point the update is provided to the wireless carriers. The delivery time for resolving an issue depends on the severity, and can be as rapid as a few days before the carrier handoff for high priority cases. The wireless carriers perform additional tests to ensure the updates will not adversely impact their networks and then plan device rollouts once that testing is complete. Carrier updates usually take at least two weeks to as much as two months (depending on the type and severity of the update) to be rolled out to customers. However, the Carriers also release monthly Maintenance Releases in order to address security-critical issues, and Samsung itself maintains a security blog (https://security.samsungmobile.com) in order to disseminate information directly to the public.

Samsung communicates with the reporting party to inform them of the status of the reported issue. Further information about updates is handled through the carrier release notes. Issues reported to Google directly are handled through Google's notification processes.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

Assurance Activities: The evaluator shall prepare a test plan and report documenting the testing aspects of the system. The test plan covers all of the testing actions contained in the [CEM] and the body of this PP's Evaluation Activities. While it is not necessary to have one test case per test listed in an evaluation activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered.

The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform. This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this PP and used by the cryptographic protocols being evaluated (IPsec, TLS/HTTPS, SSH).

The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results. The test report (which could just be an annotated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a 'fail' and 'pass' result (and the supporting details), and not just the 'pass' result.

The evaluator created a Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results.

The TOE was made available at the Gossamer testing laboratory. When performing testing, the evaluator configured the TOE into CC mode as described in the Admin Guide. The following diagram shows the evaluator test configuration:

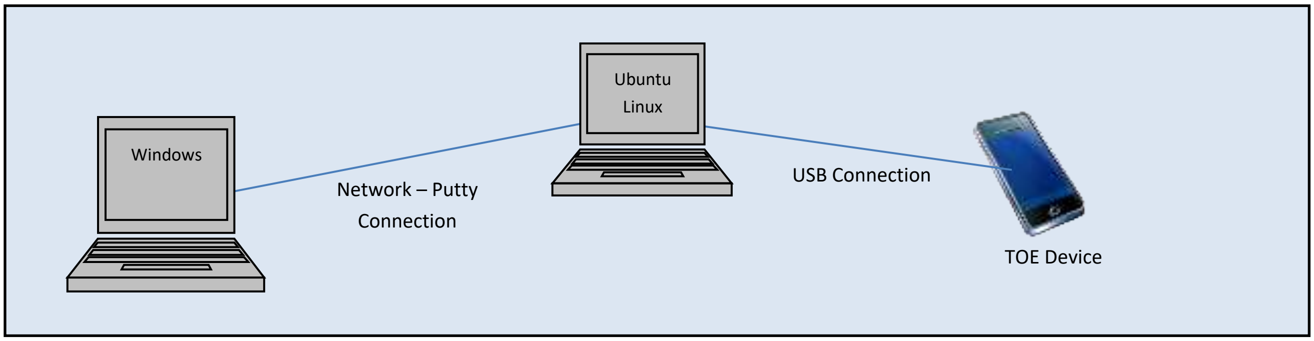

Figure 1 Evaluator Test Setup 1

The evaluator used the following test tools: Windows, Linux, Putty, Wireshark, Freeradius, OpenSSL, web server, adb, HxD, strongswan, tcpdump, micro-httpd, and Gossamer and Samsung developed test programs.

3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: The evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in mobile devices and the implemented communication protocols in general, as well as those that pertain to the particular TOE. The evaluator documents the sources consulted and the vulnerabilities found in the report.

For each vulnerability found, the evaluator either provides a rationale with respect to its nonapplicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

The vulnerability analysis is in the Detailed Test Report (DTR) prepared by the evaluator. The vulnerability analysis includes a public search for vulnerabilities using the (**NVD**) National Vulnerability Database (https://web.nvd.nist.gov/view/vuln/search) and (**VND**) Vulnerability Notes Database (http://www.kb.cert.org/vuls/) web sites https://www.cve.org/) on 12/16/2024 (from 2/1/2024) with the following search terms: "Galaxy A52", "SM-A526", "Galaxy A53", "SM-A536", "SM-S536", "Galaxy S24", "SM-S721", "Galaxy Z Flip6", "SM-F741", "Galaxy Z Fold6", "SM-F956", "SM-F958", "Knox", "BoringSSL", "strongswan", "charon", "Android", "SCrypto", "Samsung Crypto", "Samsung Kernel Crypto", "Exynos", "Qualcomm Snapdragon".

The search did not discover any known vulnerabilities in the TOE.